AD NUMBER

AD875083

NEW LIMITATION CHANGE

TO

Approved for public release, distribution unlimited

FROM

Distribution authorized to U.S. Gov't. agencies and their contractors; Critical Technology; JUN 1969. Other requests shall be referred to Dean of Engineering, Air Force Institute of technology, ATTN: AFIT-SE, Wright-Patterson AFB, Oh 45433.

AUTHORITY

AFIT memo dtd 22 Jul 1971

AIR FORCE INSTITUTE OF TECHNOLOGY

AIR UNIVERSITY
UNITED STATES AIR FORCE

APPLICATION OF THE EXTENDED KALMAN
FILTER TO BALLISTIC TRAJECTORY
ESTIMATION AND PREDICTION

THESIS

Joseph C. Orwat
1/Lt. USAF

Donald K. Potter
Civ GS-13

GGC/EE/69-[illegible]

SCHOOL OF ENGINEERING

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

91

APPLICATION OF THE EXTENDED KALMAN FILTER TO BALLISTIC TRAJECTORY ESTIMATION AND PREDICTION

THESIS

Joseph C. Orwat
1/Lt. USAF

Donald K. Potter
Civ GS-13

GGC/EE/69-15

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of the Dean of Engineering, Air Force Institued of Technology (AFIT-SE), Wright-Patterson Air Force Base, Ohio, 45433.

GGC/EE/69-15

APPLICATION OF THE EXTENDED KALMAN FILTER TO BALLISTIC TRAJECTORY ESTIMATION AND PREDICTION

THESIS

Presented to the Faculty of the School of Engineering of
the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the
Master of Science Degree
in Electrical Engineering

by

Joseph C. Orwat, B.S.E.E.
1/Lt. USAF

Donald K. Potter, B.S.E.E.
Civ USAF

Graduate Guidance and Control

June 1969

Preface

The Kalman Filter is a minimum variance filter derived with the following assumptions: the dynamics of the system are linear, the observations are linear functions of the states, and all of the noise sources and their statistical characteristics are known. For the case of estimating the state of the ballistic re-entry vehicle on the basis of noisy measurements, the Kalman theory cannot be applied directly. The validity of the linearizations made in the extension of the Kalman Filter are examined.

We wish to express our indebtedness to Lt. Col. Roger W. Johnson our thesis advisor for his continual encouragement, advice, and patience throughout this study.

Contents

	Page
Preface	ii
List of Figures	iv
List of Tables	v
Abstract	vi
I. INTRODUCTION	1
II. FILTER EQUATIONS	3
The Linear-Gaussian Case	3
Data Needed for Kalman Filter	4
Iterative Process	5
The Extended Kalman Filter	5
III. EQUATIONS FOR ESTIMATION OF A BALLISTIC TRAJECTORY	8
Coordinate System	8
Equations of Motion	8
Choice of Filter States	10
Observation Equations	12
Linearizations about Estimated Trajectory	19
Filter Equations Simplification	19
Initial Estimate of Trajectory	20
Initial State Covariance Matrix	22
Determination of Tangent-Plane Coordinate System	24
IV. SIMULATION	27
V. RESULTS	30
Bibliography	45
Appendix A: Computer Listing	46
Vita: Joseph C. Orwat	80
Vita: Donald K. Potter	81

List of Figures

Figure		Page
1	Radar Coordinate System	14
2	Aircraft Coordinate System	16
3	Tangent-Plane Coordinate System	23
4	Kalman Filter Mechanization	26
5	Simulation Model	28
6	Aircraft-Missile Configuration A	30
7	Aircraft-Missile Configuration B	31
8	Position and Prediction Errors Configuration A, $\beta = 500\ lb/ft^2$	33
9	Velocity Error Configuration A, $\beta = 500\ lb/ft^2$	34
10	Estimated Ballistic Coefficient Configuration A, $\beta = 500\ lb/ft^2$	35
11	Position and Prediction Errors Configuration B, $\beta = 500\ lb/ft^2$	36
12	Velocity Error Configuration B, $\beta = 500\ lb/ft^2$	37
13	Estimated Ballistic Coefficient Configuration B, $\beta = 500\ lb/ft^2$	38
14	Position and Prediction Errors Configuration A, $\beta = 1{,}750\ lb/ft^2$	39
15	Velocity Error Configuration A, $\beta = 1{,}750\ lb/ft^2$	40
16	Estimated Ballistic Coefficient Configuration A, $\beta = 1{,}750\ lb/ft^2$	41
17	Position and Prediction Errors Configuration B, $\beta = 1{,}750\ lb/ft^2$	42
18	Velocity Error Configuration B, $\beta = 1{,}750\ lb/ft^2$	43
19	Estimated Ballistic Coefficient Configuration B, $\beta = 1{,}750\ lb/ft^2$	44

List of Tables

Table		Page
1	Nomenclature for Vehicle Equations of Motion	10
2	Nomenclature for Kalman Filter	18

Abstract

This thesis presents the results of a study wherein the Kalman filtering technique is applied to the estimation and prediction of the trajectory of a ballistic missile from radar measurements made from an airborne radar system. Any intercept system which is to guide an anti-missile is critically dependent on these computational functions.

The Kalman Filter equations are based on a number of assumptions that are not entirely justified in actual practice. For the case of estimating the state of a ballistic re-entry vehicle on the basis of noisy measurements, the Kalman theory cannot be applied directly.

In this paper the Kalman estimator is extended to non-linear trajectory equations and unknown ballistic parameters. An estimation and prediction model is developed assuming that azimuth, elevation, range and range-rate data is provided from a phased-array radar aboard an aircraft. In order to evaluate the model, a digital computer program was developed wherein a reference trajectory for a missile is generated and this information, along with tracker aircraft position, is used by a radar model to generate airborne tracking information which is contaminated with noise. From this information the Kalman estimation and prediction model yields estimates of the present states and future states of the target. These are compared with the reference trajectory to evaluate the model.

APPLICATION OF THE EXTENDED KALMAN FILTER TO BALLISTIC TRAJECTORY ESTIMATION AND PREDICTION

I. INTRODUCTION

This study is concerned with the computational aspects of an airborne radar system which tracks re-entry vehicles. It is required that position and velocity of an incoming re-entry vehicle be determined from noisy radar data. Furthermore, it is necessary to predict the vehicle's future position on the basis of the present estimate of position and velocity. The first part of this problem is referred to as the "estimation problem", whereas the second part is referred to as the "prediction problem". A third aspect of the problem is "identification". Identification differs slightly from estimation in the sense that the imperfectly known parameters (e.g., ballistic coefficient) characterizing the signal-generating process are obtained from noisy observations, whereas previously the state variables (i.e., position and velocity coordinates) were estimated. Knowledge of the ballistic coefficient significantly enhances the quality of the prediction.

In the usual trajectory determination problem, we make discrete noisy measurements of variables related to the state of a vehicle whose motion is uniquely determined by its unknown initial state, and we ask, on the basis of noisy

measurements, for the "best" estimate of the state at any time. In a series of well-known papers (Ref 1,2,3) R.E. Kalman describes an optimal filter applicable to noisy, time-varying, linear systems. This filter, which is essentially a minimum variance linear estimator, is particularly suitable for trajectory determination problems in which estimates of state variables are desired as rapidly as possible. However, the trajectory estimation problem is nonlinear and the Kalman theory cannot be applied directly.

Although the Kalman filter is optimum only when the system differential equations and measurements are linear, it has found considerable use in estimating the state variables of a nonlinear system with measurements that are noise-corrupted nonlinear functions of state variables. This employment of the Kalman filter is frequently referred to as the "Extended Kalman Filter". It is an intuitive but frequently successful application of the Kalman filter in the absence of truly optimum filters for non-linear systems.

In brief, the Kalman Filter can be quite useful in estimating the state variables of nonlinear systems. However, more care must be exercised in checking theoretical results by means of simulation. When the Kalman Filter produces poor estimates of the states of a nonlinear system, ingenious changes can often produce a useful modified version.

II. FILTER EQUATIONS

The Linear - Gaussian Case

The Kalman Filter equations specify an estimate of the state of a linear time-varying dynamical system observed seqentially in the presence of additive white Gaussian noise. The equations used in the Kalman Filter are given below. The derivation of these eqbations can be found in numerous references (Ref 1,2). The linear system is described by

$$\dot{\underline{X}} = \underline{F}\,\underline{X} + \underline{U} \tag{1}$$

where the components of $\underline{X}$ are the states of the system; $\underline{F}$ is the system description matrix; and $\underline{U}$ is a white Gaussian noise process that may represent either actual input noise or inaccuracies in the system model. Observations represented by the vector $\underline{Z}$ are made according to

$$\underline{Z} = \underline{M}\,\underline{X} + \underline{V} \tag{2}$$

where $\underline{M}$, the measurement matrix, describes the linear combination of the state variables which comprise $\underline{Z}$ in the absence of noise, and $\underline{V}$ is a white Gaussian noise process assumed independent of $\underline{U}$. The covariances of $\underline{U}$ and $\underline{V}$ are denoted $\underline{Q}$ and $\underline{R}$ respectively, and it is assumed that an a priori estimate of states, $\hat{\underline{X}}$ has been made with error covariance $\underline{P}$.

The filtering equations may be written as a set of prediction equations

$$\underline{\hat{X}}_{k+1}(-) = \underline{\Phi}\,\underline{\hat{X}}_k(+) \qquad (3)$$

$$\underline{P}_{k+1}(-) = \underline{\Phi}\,\underline{P}_k(+)\,\underline{\Phi}^T + \underline{Q} \qquad (4)$$

which describes the behavior of the estimate and its error covariance between observations, and a set of correction equations

$$\underline{\hat{X}}\,(+) = \underline{\hat{X}}\,(-) + \underline{K}\,[\underline{Z} - \underline{M}\,\underline{\hat{X}}\,(-)] \qquad (5)$$

$$\underline{K} = \underline{P}\,(-)\,\underline{M}^T\,[\underline{M}\,\underline{P}\,(-)\,\underline{M}^T + \underline{R}]^{-1} \qquad (6)$$

$$\underline{P}\,(+) = [\underline{I} - \underline{K}\,\underline{M}]\,\underline{P}\,(-) \qquad (7)$$

which take into account the last observation $\underline{Z}$. The (-) and (+) indicate immediately prior to and after measurements, and $\underline{\Phi}$ is the state transition matrix of equation (1) given by

$$\underline{\Phi}(\Delta t) = e^{\underline{F}\Delta t} = \underline{I} + \underline{F}\,\Delta t + \frac{1}{2!}\,\underline{F}^2\,\Delta t^2 + \ldots \qquad (8)$$

<u>Data Needed for Kalman Filtering.</u> In order to employ the Kalman filtering process certain information about the system and the statistical characteristics of the input and measurement noises must be known or assumed. The following data is required before the Kalman filtering process can be initiated:

1. System description or $\underline{F}$ matrix for all values of time.

2. Sampling time Δt.

3. State transition matrix $\underline{\Phi}$ (Δt).

4. Measurement matrix $\underline{M}$.

5. Measurement noise covariance matrix $\underline{R}$.

6. Input noise covariance matrix $\underline{Q}$.

7. Initial state covariance matrix $\underline{P}_o(+)$.

8. Initial state estimate matrix $\underline{\hat{X}}_o(-)$.

Iterative Procedure. The following is the iterative procedure for processing the Kalman Filter.

1. Compute state transition matrix $\underline{\Phi}$ (Δt), Eq (8).

2. Update state covariance matrix $\underline{P}_{k+1}(-)$, Eq (4), using $\underline{\Phi}(\Delta t)$, $\underline{P}_k$ (+), and $\underline{Q}$.

3. Compute the filter gain matrix $\underline{K}$, Eq (6), using $\underline{M}$, $\underline{P}$ (-), and $\underline{R}$.

4. Compute estimate of state $\underline{\hat{X}}(+)$, Eq (5), using the observation $\underline{Z}$, $\underline{M}$, and $\underline{\hat{X}}(-)$.

5. Update the state covariance matrix P(+), Eq (7).

6. The above computational process is repeated each Δt time interval.

The Extended Kalman Filter

The Kalman filter is a minimum variance filter derived with the following assumptions:

1. The dynamics of the system are linear.

2. The observations are linear functions of the states.

3. All of the noise sources and their statistical characteristics are known.

For the case of estimating the state of a ballistic reentry vehicle on the basis of noisy measurements, the Kalman theory cannot be applied directly. The system equations governing the vehicle are highly non-linear, and the observation equation is non-linear.

If our knowledge of the system state is such that the matrices

$$\underline{F} = \frac{\partial \underline{f}}{\partial \underline{x}}\bigg|_{\hat{\underline{x}}} \tag{9}$$

$$\underline{M} = \frac{\partial \underline{m}}{\partial \underline{x}}\bigg|_{\hat{\underline{x}}} \tag{10}$$

are approximately constant over the range of uncertainty in $\hat{\underline{x}}$, then the state transition matrix, $\underline{\Phi}$, can be determined from equation (8) and the filter gain calculated using the redefined $\underline{F}$ and $\underline{M}$ matrices. It should be noted that $\underline{F}$ and $\underline{M}$ matrices computed from equations (9) and (10) can be non-linear functions of $\hat{\underline{x}}$.

These techniques are only approximate. They require that the disturbances, measurement noises, and uncertainties in the state be such a size that the higher order terms ignored in computing the error covariance are insignificant. If this condition is not satisfied, the application of the Kalman Filter to nonlinear systems may be useless. Care must be exercised in checking theoretical results by means of simulation. Because the error covariance equations

provide only an approximate evaluation of the estimation error statistics, Monte Carlo techniques are required to verify the use of the Extended Kalman Filter for nonlinear systems.

III. EQUATIONS FOR ESTIMATION OF A BALLISTIC TRAJECTORY

Coordinate System

The problem of predicting the trajectory of a ballistic vehicle can be formulated in several ways. Foremost in any formulation is the choice of a dynamically and computationally convenient frame of reference in which to perform the operations and solve the problem. A logical choice to satisfy this requirement is a reference frame which is fixed with respect to the earth. The coordinate system chosen has the origin at the center of the earth and a vertical axis passing through the point of acquisition of the target. One level axis is down-range and the other level axis is in a lateral direction. This system is essentially a tangent-plane coordinate system fixed on the acquisition point. The tangent-plane coordinate system has the advantage that two of its axes are physically oriented to be nominally in the missile flight plane. The initial covariance matrix of estimation error may be more easily defined and more generally applicable to all acquisition geometries. The main disadvantage of the tangent-plane system is that more computations are performed during filtering to place vectors on this frame. The tangent-plane coordinate system is shown in Figure 3 and discussed in more detail in this chapter.

Equations of Motion

Once a reference frame is chosen it is necessary to formulate the dynamic equations of motion for a ballistic

vehicle on these axes. The equations of motion for the vehicle in the tangent-plane coordinate system are

$$\ddot{X} = -\frac{\mu X}{R^3} - \frac{1}{2}\rho V \frac{1}{\beta}\dot{X} - 2[\omega_Y \dot{Z} - \omega_Z \dot{Y}]$$
$$- \omega_X [\omega_X X + \omega_Y Y + \omega_Z Z] + \Omega^2 X \tag{11}$$

$$\ddot{Y} = -\frac{\mu Y}{R^3} - \frac{1}{2}\rho V \frac{1}{\beta}\dot{Y} - 2[\omega_Z \dot{X} - \omega_X \dot{Z}]$$
$$- \omega_Y [\omega_X X + \omega_Y Y + \omega_Z Z] + \Omega^2 Y \tag{12}$$

$$\ddot{Z} = -\frac{\mu Z}{R^3} - \frac{1}{2}\rho V \frac{1}{\beta}\dot{Z} - 2[\omega_X \dot{Y} - \omega_Y \dot{X}]$$
$$- \omega_Z [\omega_X X + \omega_Y Y + \omega_Z Z] + \Omega^2 Z \tag{13}$$

where the symbols are defined in Table I.

The state vector has seven components:

$$\underline{X} = \begin{bmatrix} X \\ Y \\ Z \\ \dot{X} \\ \dot{Y} \\ \dot{Z} \\ 1/\beta \end{bmatrix} \tag{14}$$

TABLE I

NOMENCLATURE FOR VEHICLE EQUATIONS OF MOTION

X	Down-Range coordinate of vehicle
Y	Cross-Range coordinate of vehicle
Z	Vertical coordinate of vehicle
R	Distance from center of earth = $\sqrt{X^2+Y^2+Z^2}$
V	Speed of vehicle = $\sqrt{\dot{X}^2 + \dot{Y}^2 + \dot{Z}^2}$
β	Ballistic coefficient of vehicle = $\frac{W}{C_D A}$
ρ	Atmospheric density
μ	Gravitational constant
Ω	Earth rate
$\omega_x, \omega_y, \omega_z$	Tangent-plane components of earth rate

Choice of Filter States

Once the linearized model is determined, it is necessary to choose what quantities are to be estimated by the filter. Since the errors in the states of a nonlinear system behave much more linearly than the states themselves, it was decided to apply the linear filter theory only to the estimates of the errors in the states. Thus it is necessary to formulate a linearized error model which is based on the partial derivatives of the equations of motion with respect to all state variables. It is this error model which is implemented in the Kalman Filter. The state vector for the Kalman Filter is then defined as

The nonlinear system equations are then rewritten as

$$\underline{\dot{X}} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ -\frac{\mu}{R^3} X - \frac{\rho V}{2\beta}\dot{X} - 2[\omega_Y \dot{Z} - \omega_Z \dot{Y}] \\ - \omega_X [\omega_X X + \omega_Y Y + \omega_Z Z] + \Omega^2 X \\ -\frac{\mu}{R^3} Y - \frac{\rho V}{2\beta}\dot{Y} - 2[\omega_Z \dot{X} - \omega_X \dot{Z}] \\ - \omega_Y [\omega_X X + \omega_Y Y + \omega_Z Z] + \Omega^2 Y \\ -\frac{\mu}{R^3} Z - \frac{\rho V}{2\beta}\dot{Z} - 2[\omega_X \dot{Y} - \omega_Y \dot{X}] \\ - \omega_Z [\omega_X X + \omega_Y Y + \omega_Z Z] + \Omega^2 Z \end{bmatrix} \qquad (15)$$

The extended Kalman Filter equations are applied by setting

$$\underline{F} = \frac{\partial f_i}{\partial x_j} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ f_{XX} & f_{XY} & f_{XZ} & f_{X\dot{X}} & f_{X\dot{Y}} & f_{X\dot{Z}} & f_{X\beta} \\ f_{YX} & f_{YY} & f_{YZ} & f_{Y\dot{X}} & f_{Y\dot{Y}} & f_{Y\dot{Z}} & f_{Y\beta} \\ f_{ZX} & f_{ZY} & f_{ZZ} & f_{Z\dot{X}} & f_{Z\dot{Y}} & f_{Z\dot{Z}} & f_{Z\beta} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (16)$$

$$\underline{X} = \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \\ \Delta\dot{X} \\ \Delta\dot{Y} \\ \Delta\dot{Z} \\ \Delta 1/\beta \end{bmatrix} \tag{17}$$

The differential equation for these error quantities can then be written in matrix form as

$$\dot{\underline{X}} = \underline{F}\,\underline{X} \tag{18}$$

where $\underline{F}$ was defined by equation (16). It should be noted that although this is an error model, the system description matrix, $\underline{F}$, the state transition matrix, Φ, and the observation matrix, $\underline{M}$, are functions of the total estimated states. The total estimated states are determined by numerically integrating the nonlinear equations of motion and subtracting out the estimated error. Thus the total states are being "controlled".

This is the fundamental difference between applying the filter to a linear system and to the deviations of a nonlinear system.

Observation Equations

Observations of the re-entry vehicle are made every Δt seconds by means of a phased-array radar. It is now necessary to decide which quantities will be treated as observables. Measurements are made of the azimuth, A; elevation,

E; range, R; and range-rate, $\dot{R}$ (doppler velocity) of the re-entry vehicle with respect to the aircraft coordinate system. Figure 1 shows the geometry and gives the relationship between the radar and the aircraft coordinate systems.

Since the filter is being mechanized as an error model, it is necessary to treat errors in the observations as the measurements. Thus the "measurements" for the Kalman Filter are actually <u>differences</u> between system-indicated and measured position and range-rate.

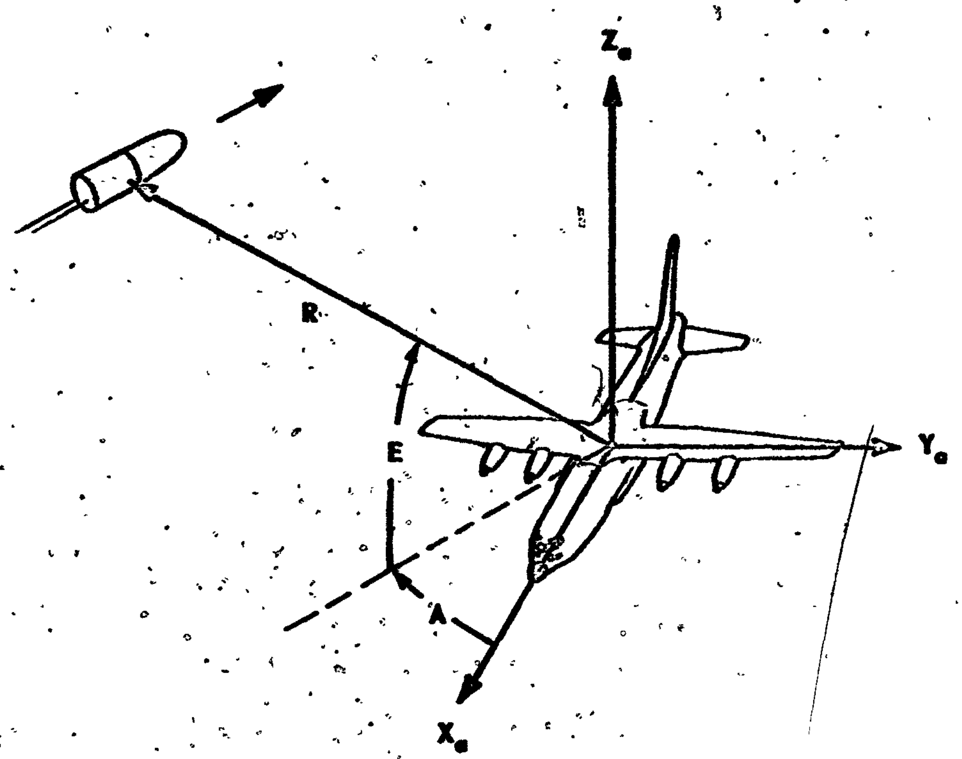
If the measurement is not given directly in the computational coordinates, it must be properly transformed through knowledge of the particular geometry involved. The transformation can either be performed outside the Kalman Filter or take place in the measurement matrix, $\underline{M}$.

The vector of observables was chosen to be

$$\underline{Z} = \begin{bmatrix} X_c - X_o \\ Y_c - Y_o \\ Z_c - Z_o \\ \dot{R}_c - \dot{R}_o \end{bmatrix} = \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \\ \Delta \dot{R} \end{bmatrix} \tag{19}$$

where the subscripts "c" and "o" refer to computed and observed quantities respectively. The measurement matrix, $\underline{M}$, is thus defined as

$$\underline{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{XR} & C_{YR} & C_{ZR} & 0 \end{bmatrix} \tag{20}$$



$X_e = R \cos E \cos A$
$Y_e = -R \cos E \sin A$
$Z_e = R \sin E + R_E + h$

Figure 1 RADAR COORDINATE SYSTEM

where the three non-zero elements in the last row are the direction cosines between the X, Y, Z tangent-plane axes and the radar line-of-sight.

Then

$$\underline{Z} = \underline{M}\,\underline{X} + \underline{W} \tag{21}$$

where $\underline{W}$ is a vector of white measurement noises.

The measurement noise covariance matrix, $\underline{R}$, is functionally dependent on the statistics of the sensor errors and the orientation of the sensor. Since the $\underline{Z}$ vector was chosen to be the three position errors and range-rate error, it is necessary to transform the noise errors of azimuth, elevation, and range into noise in the three position errors.

The relationship between the position vector of the re-entry vehicle in radar coordinates is given by

$$\begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix} = \begin{bmatrix} \cos E \cos A \\ -\cos E \sin A \\ \sin E \end{bmatrix} \begin{bmatrix} R \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ R_E + h \end{bmatrix} \tag{22}$$

Taking the differential of equation (22) yields

$$\begin{bmatrix} \Delta X_a \\ \Delta Y_a \\ \Delta Z_a \end{bmatrix} = \begin{bmatrix} -R \cos E \sin A & -R \sin E \cos A & \cos E \sin A \\ -R \cos E \cos A & R \sin E \sin A & -\cos E \sin A \\ 0 & R \cos E & \sin E \end{bmatrix} \begin{bmatrix} \Delta A \\ \Delta E \\ \Delta R \end{bmatrix} \tag{23}$$

Equation (23) is defined as

$$\Delta \underline{X}_a = \underline{A}\, \Delta \underline{V} \tag{24}$$

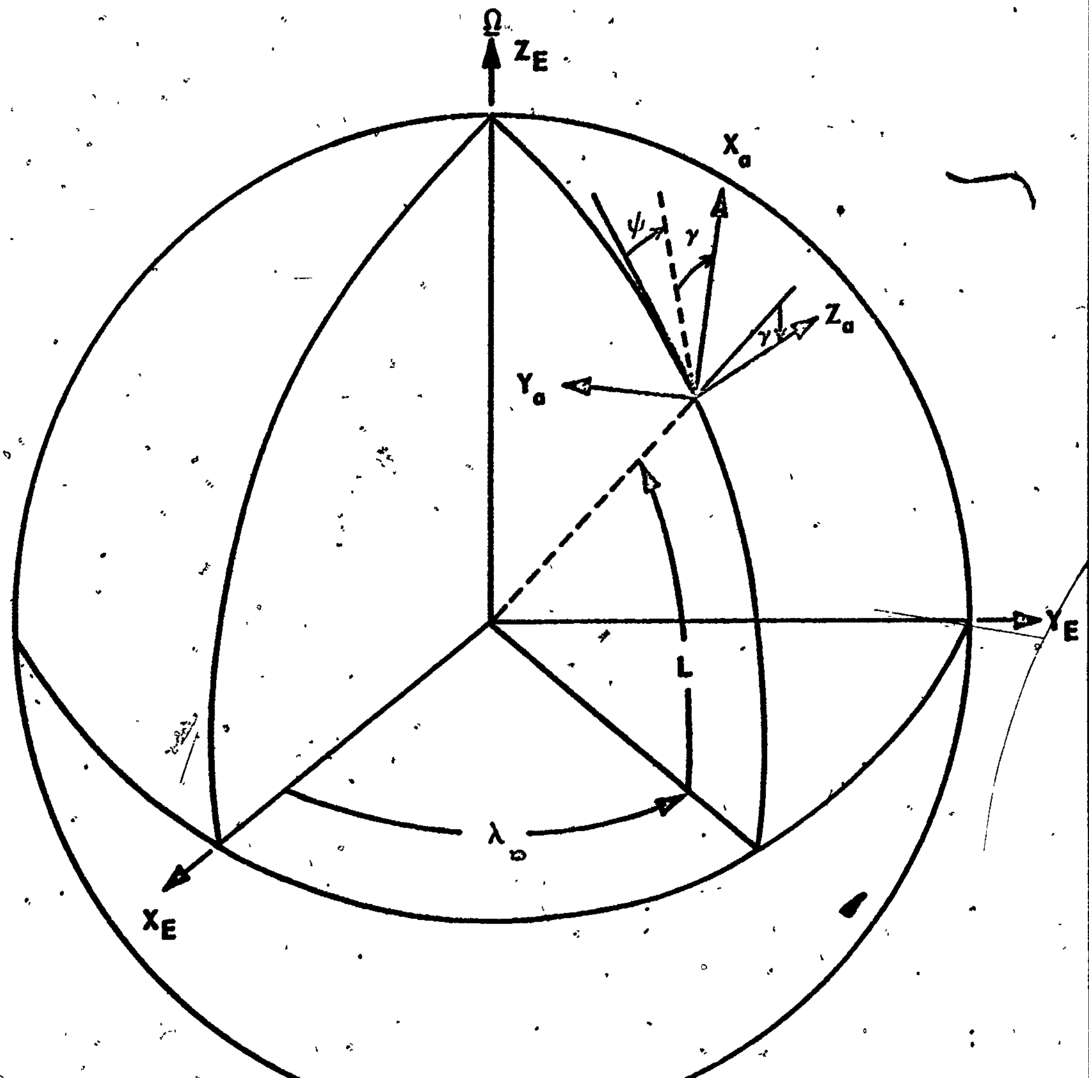


Figure 2 AIRCRAFT COORDINATE SYSTEM

Now, we see that

$$\underline{W}_1 = \underline{C}_E^T \, \underline{C}_A^E \, \underline{A} \, \underline{V}_1 \tag{25}$$

where $\underline{W}_1$ is the three position components of the measurement noise vector, $\underline{V}_1$ is noise in the radar position measurements, $\underline{C}_A^E$ is the direction cosine matrix from aircraft coordinates to earth coordinates, and $\underline{C}_E^T$ is the direction cosine matrix from the earth coordinates to the tangent-plane coordinate system. The covariance matrix of the position components of the measurement noise, denoted $\underline{R}'$, becomes

$$\underline{R}' = E\,[\underline{W}_1 \, \underline{W}_1^T] = [\underline{C}_E^T \, \underline{C}_A^E \, \underline{A}] \, \underline{R}'' \, [\underline{C}_E^T \, \underline{C}_A^E \, \underline{A}]^T \tag{26}$$

where

$$\underline{R}'' = E\,[\underline{V}_1 \, \underline{V}_1^T] = \begin{bmatrix} \sigma_A^2 & 0 & 0 \\ 0 & \sigma_E^2 & 0 \\ 0 & 0 & \sigma_R^2 \end{bmatrix} \tag{27}$$

The total covariance matrix for measurement noise has the form

$$\underline{R} = \begin{bmatrix} R'_{11} & R'_{12} & R'_{13} & 0 \\ R'_{21} & R'_{22} & R'_{23} & 0 \\ R'_{31} & R'_{32} & R'_{33} & 0 \\ 0 & 0 & 0 & \sigma_{\dot{R}}^2 \end{bmatrix} \tag{28}$$

TABLE II

NOMECLATURE FOR KALMAN FILTER

ΔX - Down-range position error of vehicle

ΔY - Cross range position error of vehicle

ΔZ - Vertical position error of vehicle

A - Azimuth angle of vehicle relative to aircraft

E - Elevation angle of vehicle relative to aircraft

R - Range from aircraft to vehicle

λ - Aircraft longitude

L - Aircraft latitude

Ψ - Aircraft heading

γ - Aircraft flight-path angle

h - Aircraft altitude

R_E - Radius of earth

C_A^E - Aircraft-to-earth transformation

C_E^T - Earth-to-tangent-plane transformation

C_{XR}, C_{YR}, C_{ZR} - Direction cosines between X, Y, Z axis and radar line-of-sight

$\underline{F}$ - System Description Matrix

$\underline{\Phi}$ - State Transition Matrix

$\underline{M}$ - Measurement Matrix

$\underline{K}$ - Filter coefficients Matrix

$\underline{P}$ - State covariance Matrix

$\underline{Q}$ - Input noise covariance Matrix

$\underline{R}$ - Measurement noise covariance Matrix

Linearization About Estimated Trajectory

So far it has been assumed that a nominal trajectory is available for linearization purposes. A procedure similar to that suggested by Schmidt (Ref 4) is used to eliminate the need for the assumed trajectory. As mentioned previously, the total states are being controlled. The total estimated states are determined by numerically integrating the non-linear equations of motion and subtracting out the estimated error. The control equation is

$$\hat{\underline{X}}(+) = \hat{\underline{X}}(-) - \hat{\underline{x}} \tag{29}$$

where $\hat{\underline{X}}$ contains the estimates of the total states and $\hat{\underline{x}}$, the errors in the states. Thus, we are always linearizing about our estimated trajectory. This could cause large errors initially in the linearity assumptions since the initial estimated trajectory could be way off. However, the estimates improve rapidly and the assumptions become valid.

Filter Equations Simplification

Not only does this technique provide a good "nominal" trajectory to linearize about, but it also provides a simplification of the Kalman Filter equations. Equation (5) can be written as

$$\hat{\underline{x}}_{n+1} = \underline{\Phi}_n \hat{\underline{x}}_n + \underline{K}_{n+1}[\underline{Z}_{n+1} - \underline{M}_{n+1} \underline{\Phi}_n \hat{\underline{x}}_n] \tag{30}$$

Since the total variables are now being controlled in addition to being estimated,

$$\hat{\underline{X}}_n = 0 \tag{31}$$

Immediately after the measurements are made, the next estimate of the system errors is given by

$$\hat{\underline{X}}_{n+1} = \underline{K}_{n+1} \underline{Z}_{n+1} \tag{32}$$

The simplification eliminates the need to compute $\underline{\Phi}_n \underline{X}_n$ and $\underline{M}_{n+1} \underline{\Phi}_n \underline{X}_n$. The matrices $\underline{\Phi}_n$ and $\underline{M}_{n+1}$ are, however, still required for the calculation of $\underline{K}_{n+1}$.

This completes the necessary equations for implementation of the Extended Kalman Filter. We must determine the initial values for the estimated trajectory $\hat{\underline{X}}_o$ and values for the initial state covariance matrix $\underline{P}_o$, as well as define the tangent-plane coordinate system which is the computational frame for filter mechanization.

Initial Estimate Of Trajectory

To apply the Kalman Filter, an initial estimate of the state of the nonlinear system and the covariance matrix of errors in this estimate must be available. A reasonable way of obtaining this is by use of the least-squares fit to a polynomial. The coefficients of a second order polynomial were determined by

$$\begin{bmatrix} \hat{a}_o \\ \hat{a}_1 \\ \hat{a}_2 \end{bmatrix} = \begin{bmatrix} N & \Sigma t_i & \Sigma t_i^2 \\ \Sigma t_i & \Sigma t_i^2 & \Sigma t_i^3 \\ \Sigma t_i^2 & \Sigma t_i^3 & \Sigma t_i^4 \end{bmatrix}^{-1} \begin{bmatrix} \Sigma X \\ \Sigma X t_i \\ \Sigma X t_i^2 \end{bmatrix} \tag{33}$$

where the summations are from 1 to N. Coefficients of Y and Z were obtained similarly. Note the inverted matrix is the same for all three cases. The values of X, Y and Z are the components of the position vector from the aircraft to the vehicle expressed in earth coordinates by rotating the vector through the aircraft-to-earth direction cosines C_A^E. Thus the polynomial fit is applied to the three earth components of the vehicle trajectory.

The vehicle is nominally tracked for four seconds before the coefficients of the least-squares polynomial fit are calculated. Then, estimated position vectors of the vehicle in earth coordinates are calculated for time equal to zero and time equal to four seconds by

$$\begin{aligned} \hat{X}(t) &= \hat{a}_o + \hat{a}_1 t + \hat{a}_2 t^2 \\ \hat{Y}(t) &= \hat{b}_o + \hat{b}_1 t + \hat{b}_2 t^2 \\ \hat{Z}(t) &= \hat{c}_o + \hat{c}_1 t + \hat{c}_2 (t)^2 \\ \hat{R}(t) &= \sqrt{\hat{X}(t)^2 + \hat{Y}(t)^2 + \hat{Z}(t)^2} \end{aligned} \tag{34}$$

These two position vectors are used to establish the tangent-plane coordinate system and the direction cosines from earth-to-tangent-plane, C_E^T are calculated. A

velocity estimate at time equal to four seconds is calculated by

$$\hat{\dot{X}}(t) = \hat{a}_1 + 2\,\hat{a}_2 t$$
$$\hat{\dot{Y}}(t) = \hat{b}_1 + 2\,\hat{b}_2 t \qquad (35)$$
$$\hat{\dot{Z}}(t) = \hat{c}_1 + 2\,\hat{c}_2 t$$

where these equations are the time derivatives of the polynomials in equation (34). The components of position and velocity are then rotated into the tangent-plane system and become the initial conditions of the estimated states for the start of Kalman Filtering.

Initial State Covariance Matrix

A technique exists whereby the covariance matrix for the estimated states can be determined from the variances assumed for the radar system (Ref 6). However, these estimates are not critical to the process so long as they are not grossly underestimated. Studies show that it is better to overestimate the error for self-correlation terms rather than to underestimate, whereas, it is better to underestimate the cross-correlated terms. Thus, we choose to set all cross-correlation terms equal to zero, and calculate the diagonal terms by

$$P_{11} = P_{22} = P_{33} = (R\sigma_E)^2$$
$$P_{44} = P_{55} = P_{66} = \left(\frac{R\sigma_E}{\Delta t}\right)^2 \qquad (36)$$
$$P_{77} \text{ - read as input data}$$

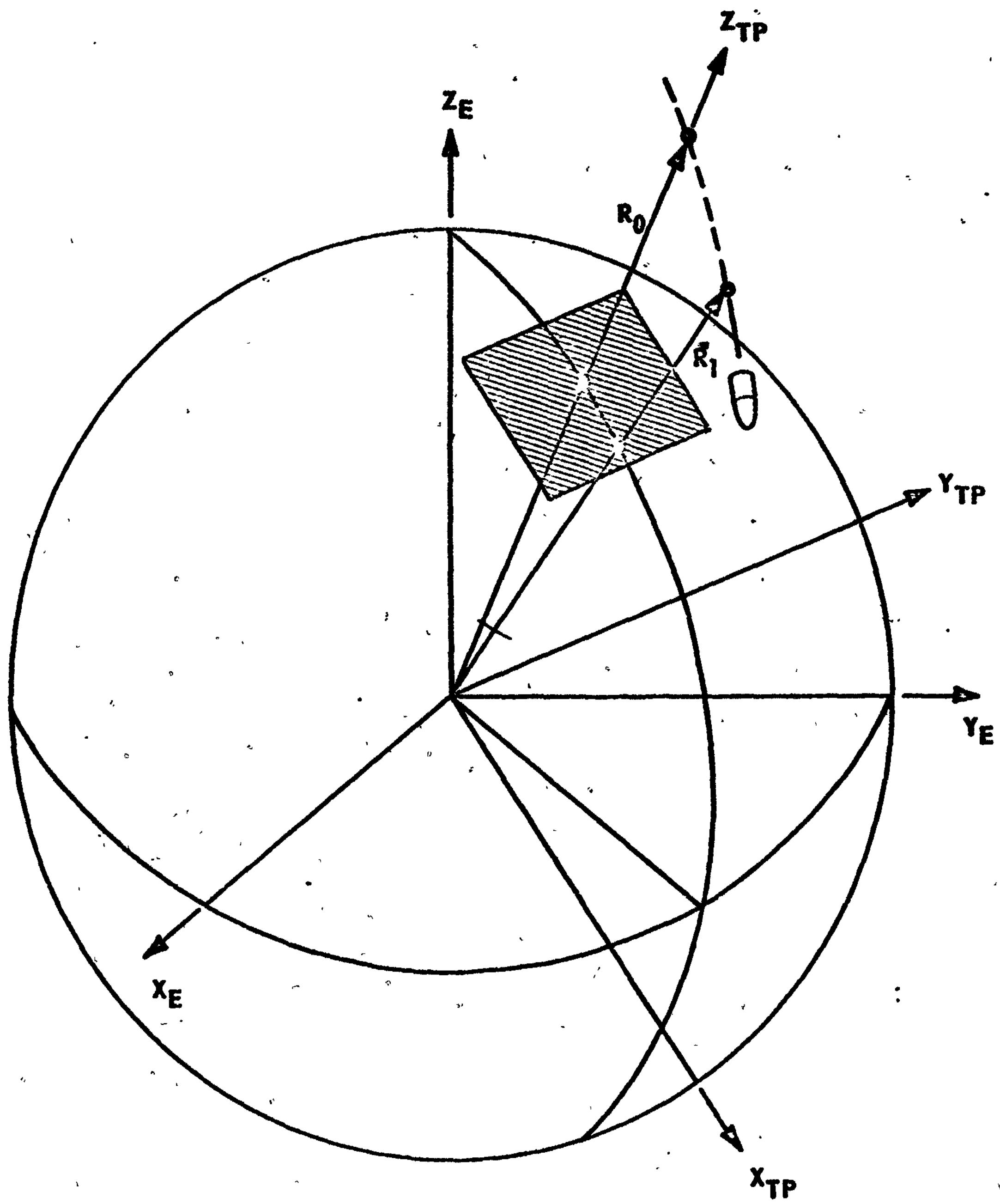


Figure 3 TANGENT PLANE COORDINATE SYSTEM

where R is the range of the vehicle from the aircraft, σ_E is the rms value of elevation angle error of the vehicle, and Δt is the tracking time for the least-squares fit. Elevation error was chosen because it is generally larger than azimuth error. This technique has proved to estimate position error about 50 percent high and velocity error about 100 percent high when compared to the fitted error for the geometries and radar errors considered.

These initial guesses could use some refinement since our studies have shown the dynamic response of the filter to be a function of $\underset{\sim}{P}_o$.

Determination Of Tangent-Plane Coordinate System

In the analysis, radar measurements were collected nominally for four seconds. This data was used to form preliminary least-squares curve fits to the trajectory for the purpose of obtaining initial position of the vehicle at acquisition and acquisition plus four seconds, as described previously. Denoting the position vectors, in earth coordinates, at times zero and four seconds, as $\underset{\sim}{R}_o$ and $\underset{\sim}{R}_1$ respectively, the product

$$\frac{\underset{\sim}{R}_o \times \underset{\sim}{R}_1}{\left|\underset{\sim}{R}_o \times \underset{\sim}{R}_1\right|} = \underset{\sim}{i}_\eta \tag{37}$$

defines the unit vector which is normal to the trajectory plane and along the Y_{TP} axis as shown in Figure 3. The product

$$\frac{\underline{i}_\eta \times \underline{R}_o}{|\underline{R}_o|} = \underline{i}_\delta \tag{38}$$

defines the unit vector which is down-range and along the X_{TP} axis. The unit vector in the vertical direction is simply

$$\frac{\underline{R}_o}{|\underline{R}_o|} = \underline{i}_\nu \tag{39}$$

Thus, the tangent-plane coordinate system, which is the computational frame for the Kalman Filter, has been established.

Components of these vectors on earth coordinates from a direction cosine matrix C_E^T between the earth and the tangent-plane coordinate systems, where

$$\underline{C}_E^T = \begin{bmatrix} i_{\delta X} & i_{\delta Y} & i_{\delta Z} \\ i_{\eta X} & i_{\eta Y} & i_{\eta Z} \\ i_{\nu X} & i_{\nu Y} & i_{\nu Z} \end{bmatrix} \tag{40}$$

The transformation between aircraft and tangent-plane is simply

$$\underline{C}_A^T = \underline{C}_E^T \, \underline{C}_A^E \tag{41}$$

Any inversion transformation is simply the transpose since direction cosine matrices are orthonormal.

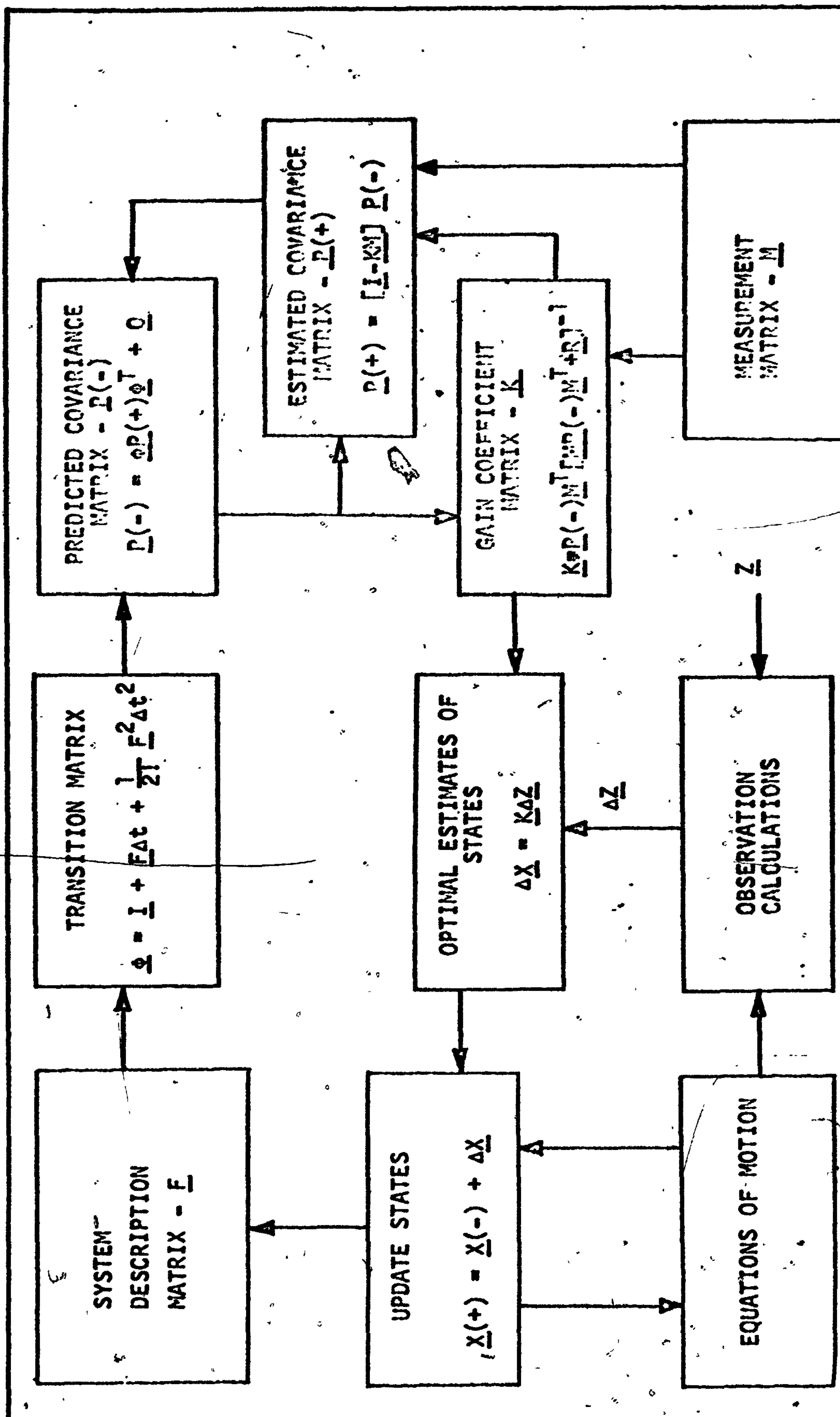


Figure 4 Kalman Filter Mechanization

IV. SIMULATION

A computer program is implemented to evaluate the Kalman Filter. An airborne radar platform is simulated to provide tracking data. A radar model and an aircraft model are used to simulate the airborne radar platform. Altitude, velocity, heading, latitude, and longitude describe the initial flight conditions of the aircraft. Azimuth, elevation, range, and range-rate from the aircraft to the reference trajectory are calculated by use of the radar model. Noise is added to the radar information to corrupt these perfect measurements. A noise model is used to provide zero mean Gaussian noise for any specified standard deviation. By also specifying an auto-correlation time constant, it can produce "exponentially correlated noise" (Ref 7).

An Adams-Moulton, Adams-Bashford predictor-corrector method is used to integrate the non-linear equations of motion for the reference trajectory. A Runge-Kutta method is used to integrate the Tangent-Plane Kalman Filter trajectory. The error estimates from the Kalman Filter model are subtracted from the non-linear equations of motion to give the best estimate of the position, velocity, and ballistic coefficient of the ballistic missile.

The prime element in any intercept problem is the ability to accurately predict the position of the missile at some future time. This prediction is accomplished by integrating the equations of motion, using as initial conditions

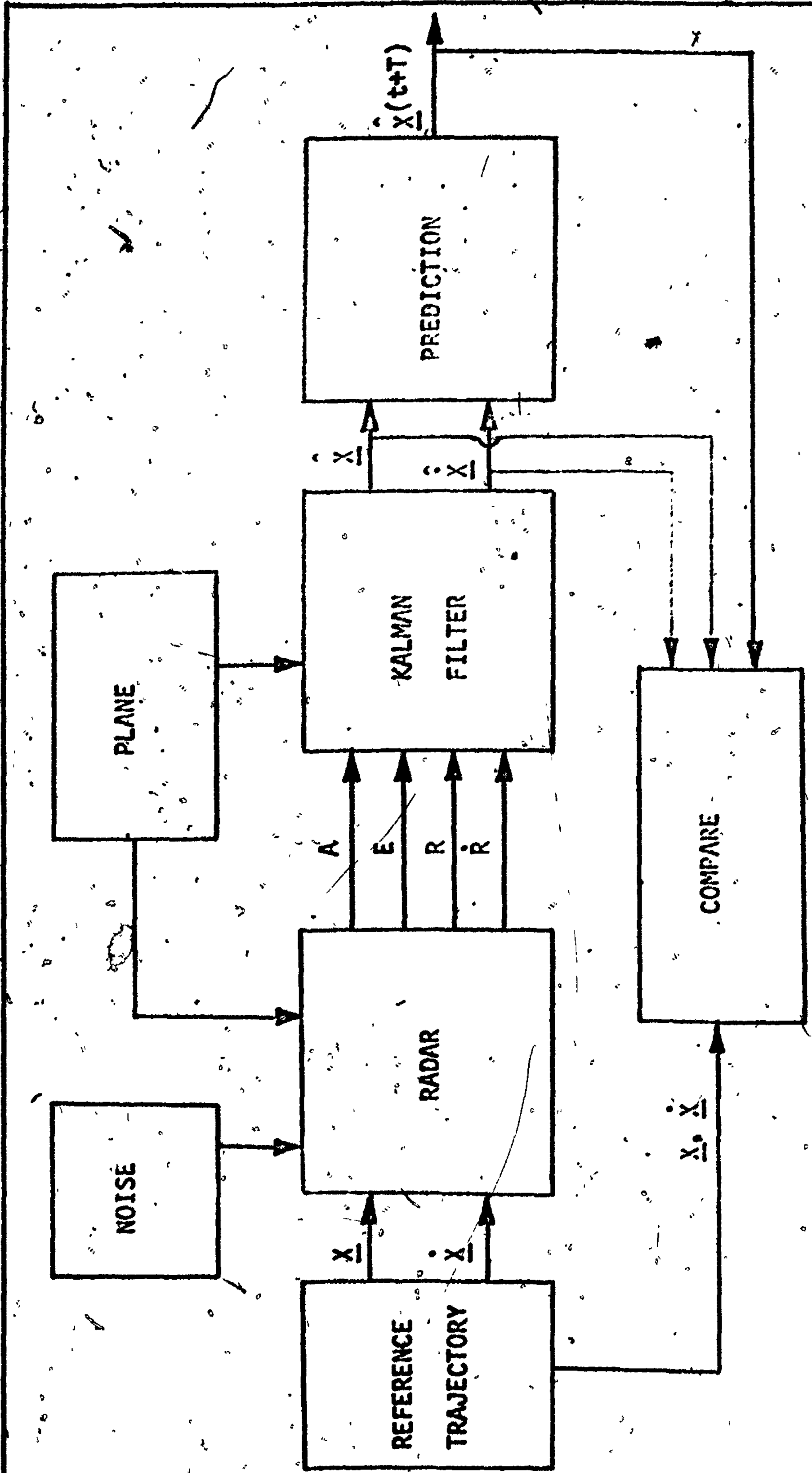


Figure 5 Simulation Model

the non-linear states that are corrected by the Kalman filter error estimates. The prediction result is evaluated by comparing it to the reference trajectory (Figure 8 through Figure 19).

V. RESULTS

Four trajectories are used to evaluate the Kalman filter: two aircraft missile configurations in combination with high and low ballistic coefficients. Configuration A was constructed so that the vehicle flew past the aircraft (Figure 6). This configuration allows us to investigate the effect of having no velocity information about the missile (zero range-rate) during part of the tracking period. This

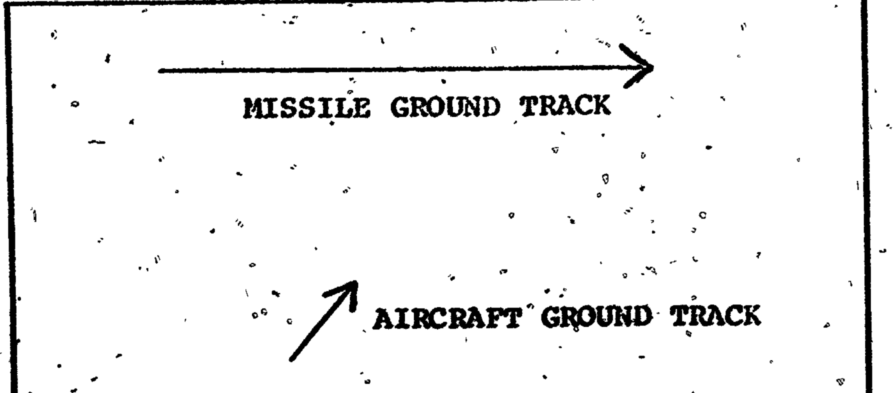


Figure 6 Aircraft-Missile Configuration A

occurs when the distance between the aircraft and the missile is at a minimum.

Configuration B is constructed so that the vehicle is always approaching the aircraft (Figure 7). This configuration allows us to investigate the effect of having non-zero range-rate information for the entire period of observation.

	Configuration A Figures	Configuration B Figures
$\beta = 500\ lb/ft^2$	8, 9, 10	11, 12, 13
$\beta = 1,750\ lb/ft^2$	14, 15, 16	17, 18, 19

The position errors, Figures (8,11,14,17) show the actual position errors between the reference trajectory and the estimated trajectory. Also, three plots of position prediction error are shown as prediction was started with the

Figure 7 Aircraft-Missile Configuration B

information available after ten seconds, twenty seconds, and thirty seconds of processing data through the Kalman filter. One would expect better prediction results after more data has been processed. However, by inspection of position prediction errors for Configuration A, Figure (8) and Figure (14), this is not always the case. In order to explain the effect of increased position prediction error after more data has been processed, the velocity errors and the estimated ballistic coefficient must be examined at the start of prediction. In either the high or low ballistic coefficient case, the velocity error decreases at first, then increases, and finally decreases again. During the period of the first decrease, the missile is above the atmosphere and an incorrect estimated ballistic coefficient has no effect on the trajectory. As the missile enters the atmosphere with an incorrectly estimated ballistic coefficient, the velocity error starts to increase due to the functional relationship

between the velocity of the missile and its ballistic coefficient. Also, during the period of increasing velocity error, the range-rate is approaching zero as the range from the aircraft to the missile approaches a minimum. As more data is processed through the Kalman filter, the estimated value of the ballistic coefficient nears its actual value and the velocity error decreases.

For aircraft-missile Configuration B, there is an expected asympotic decrease in the velocity error, due to the availability of non-zero range-rate during the entire tracking period. However prediction errors have not significantly improved over Configuration A because during prediction the value of the estimated ballistic coefficient is incorrect. The prediction errors do not increase as rapidly at the start of prediction as in Configuration A, but still do increase. The delay in the error build-up is due to the small velocity error at the start of prediction. However as prediction continues an incorrectly estimated ballistic coefficient causes the velocity error to increase rapidly thereby increasing the position errors also. One may conclude that no matter how accurate the position and velocity of the missile is known at the start of prediction, the prime element in the prediction problem is the ballistic coefficient. In order to arrive at any firm conclusions a parametric study must be made; such as, accuracy as a function of tracking time, tracking geometry, and a priori information.

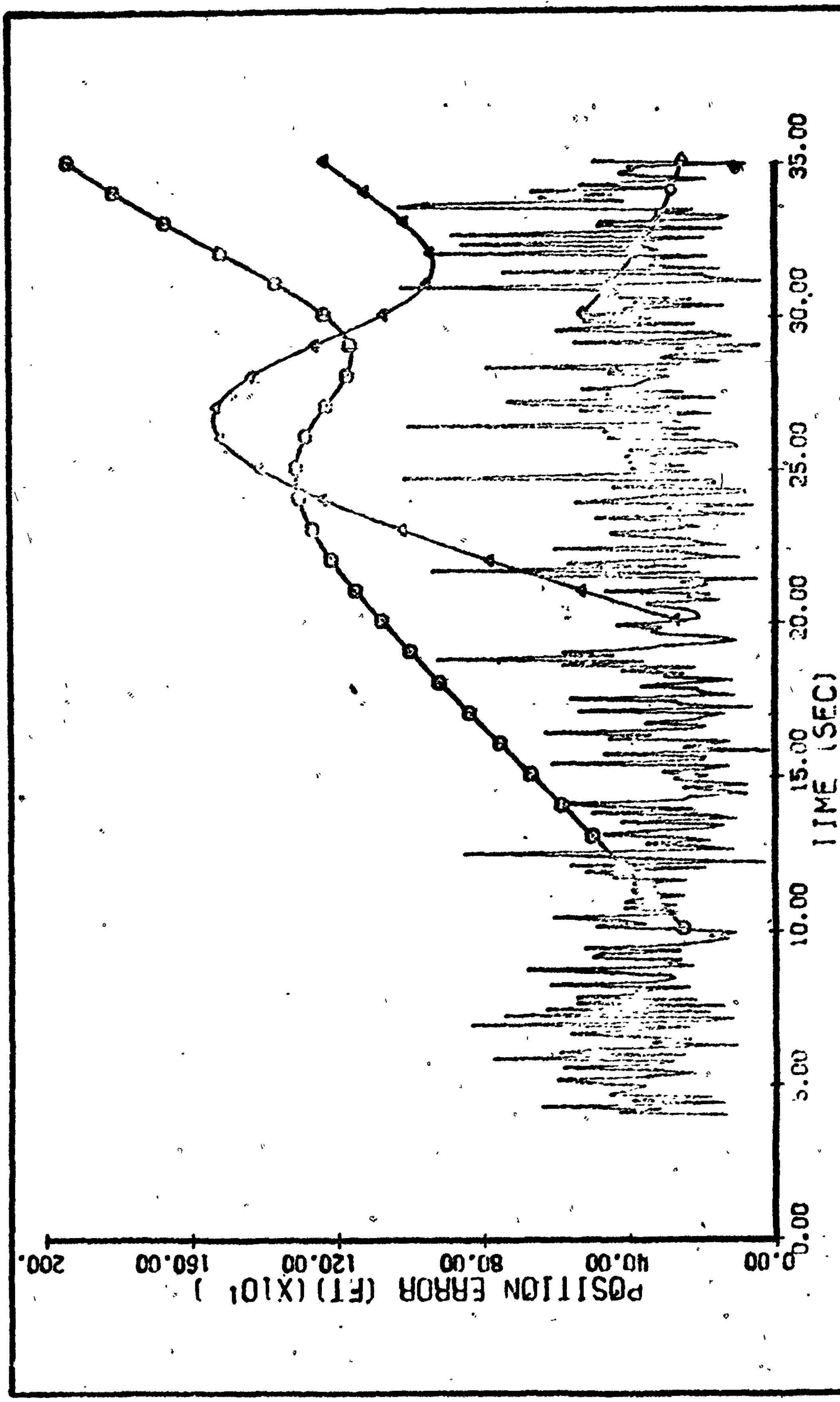


Figure 8 Position and Prediction Errors

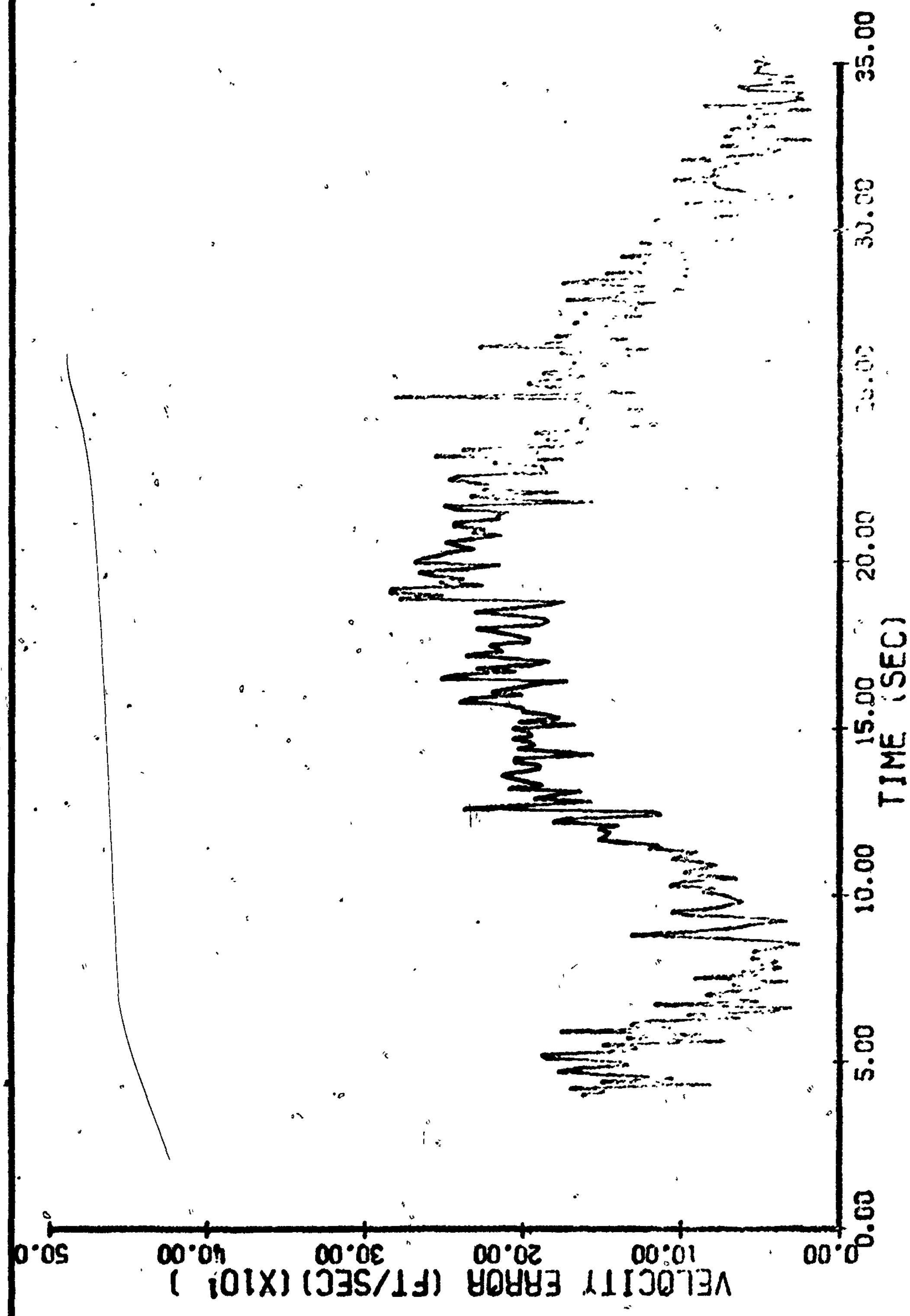


Figure 9 Velocity Error

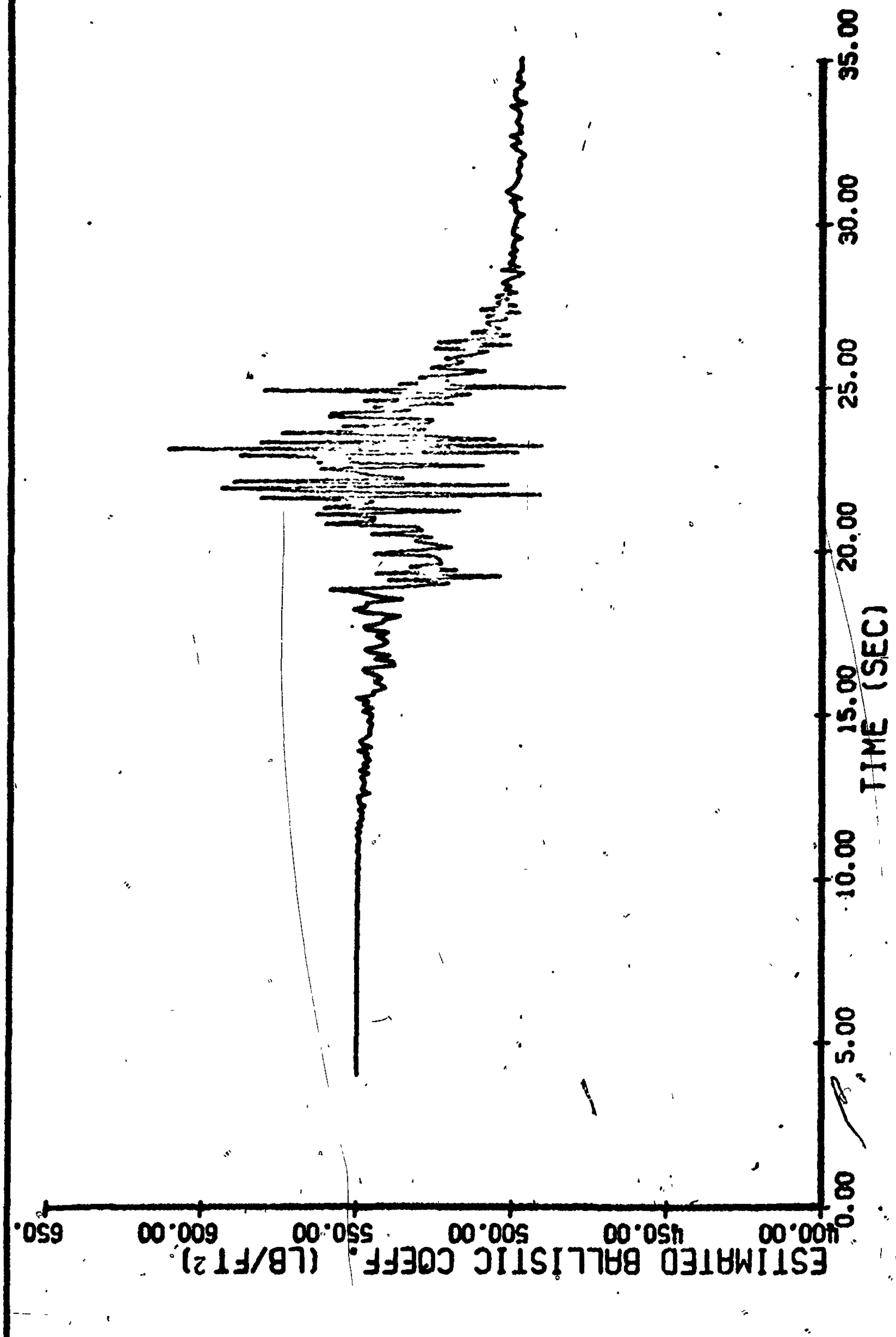


Figure 10 Estimated Ballistic Coefficient

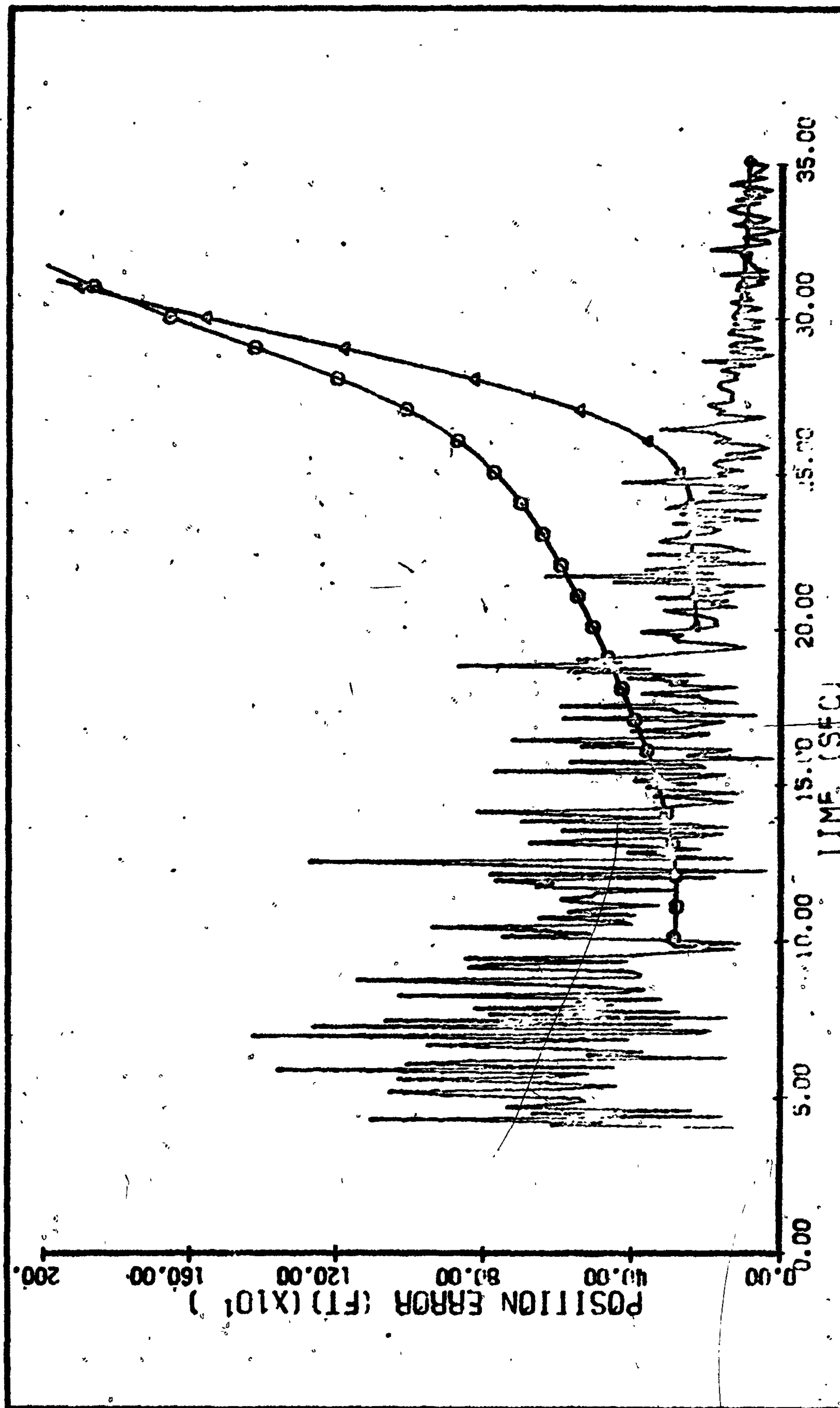


Figure 11 Position and Prediction Errors

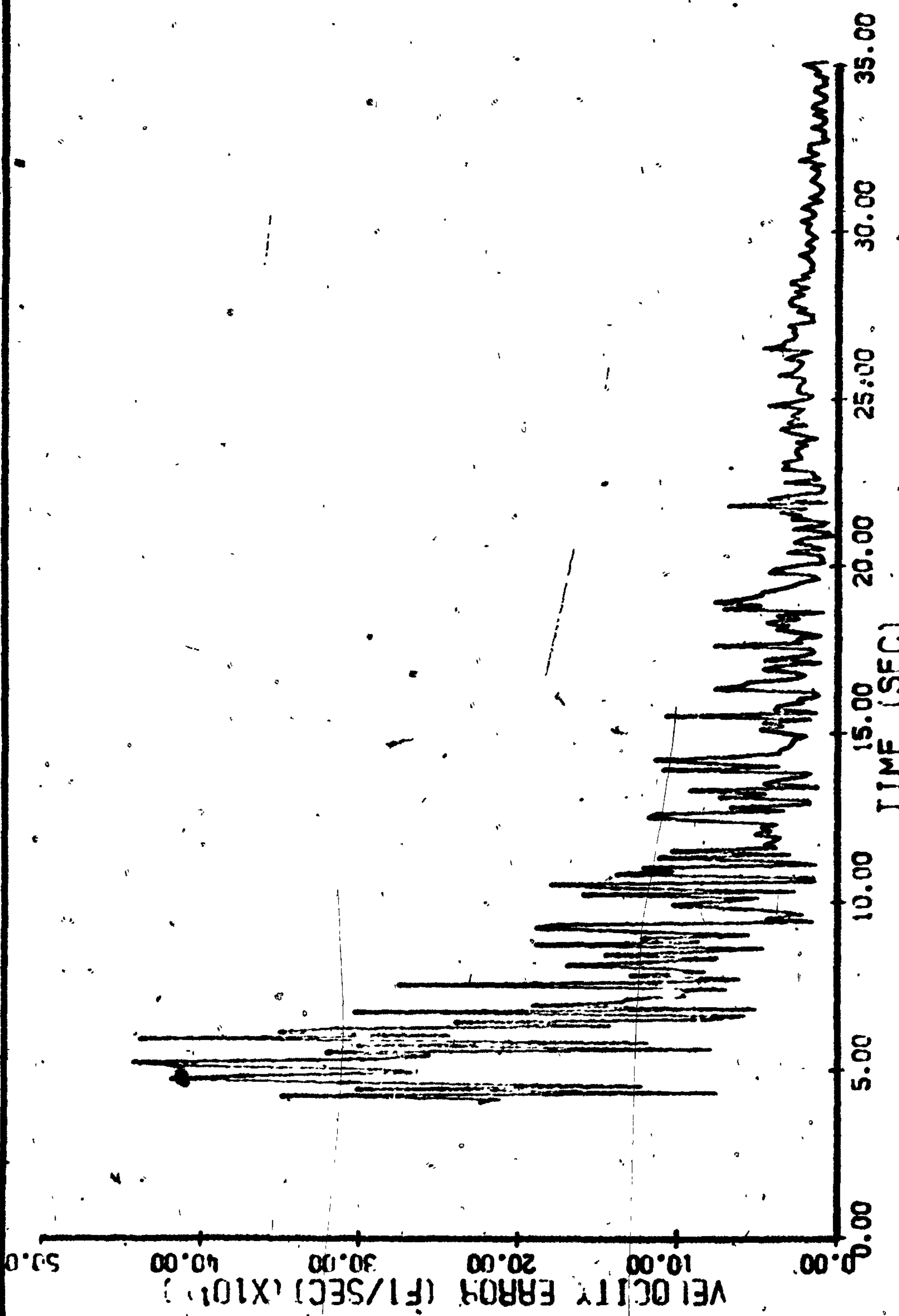


Figure 12 Velocity Error

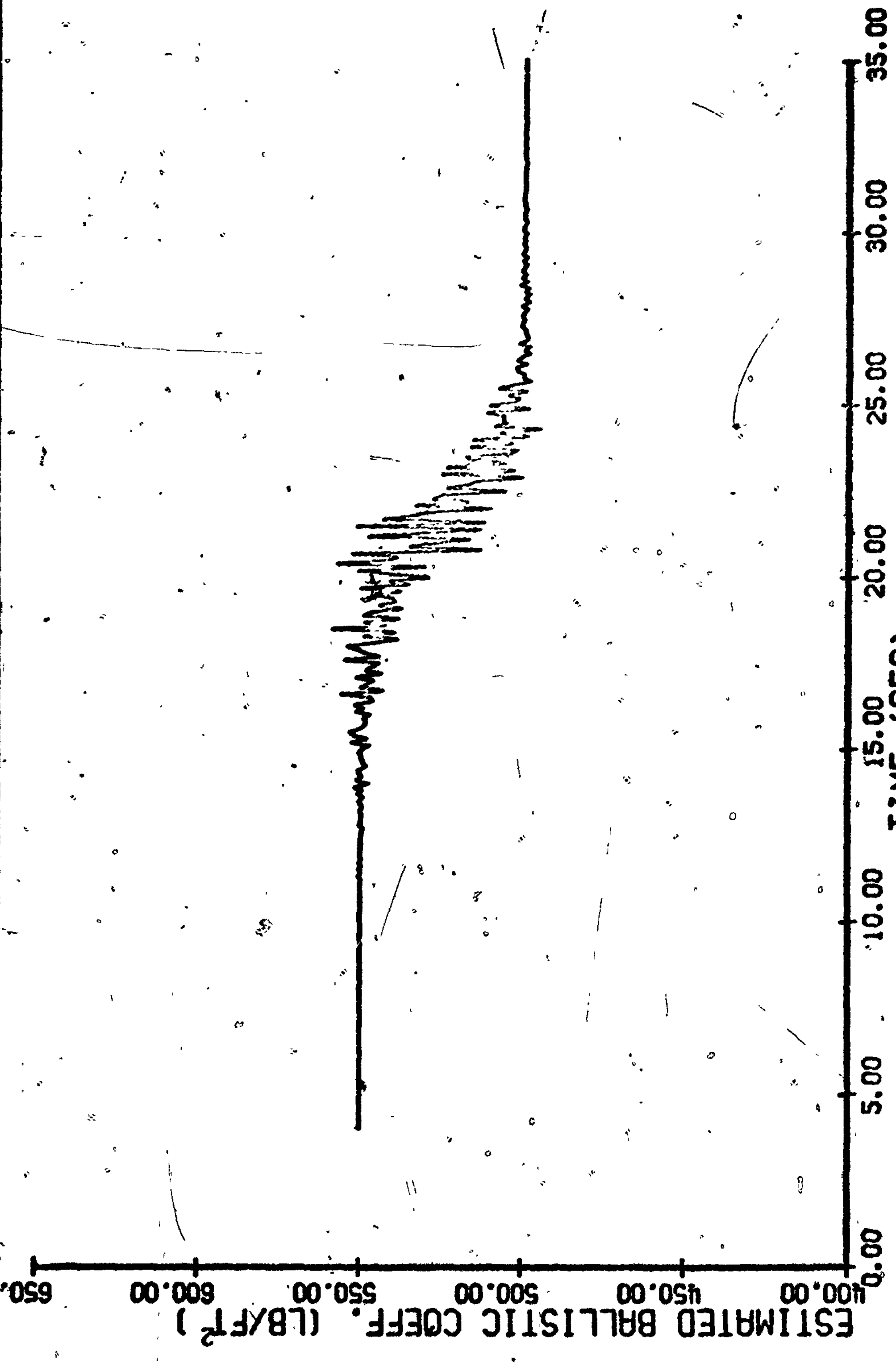


Figure 13 Estimated Ballistic Coefficient

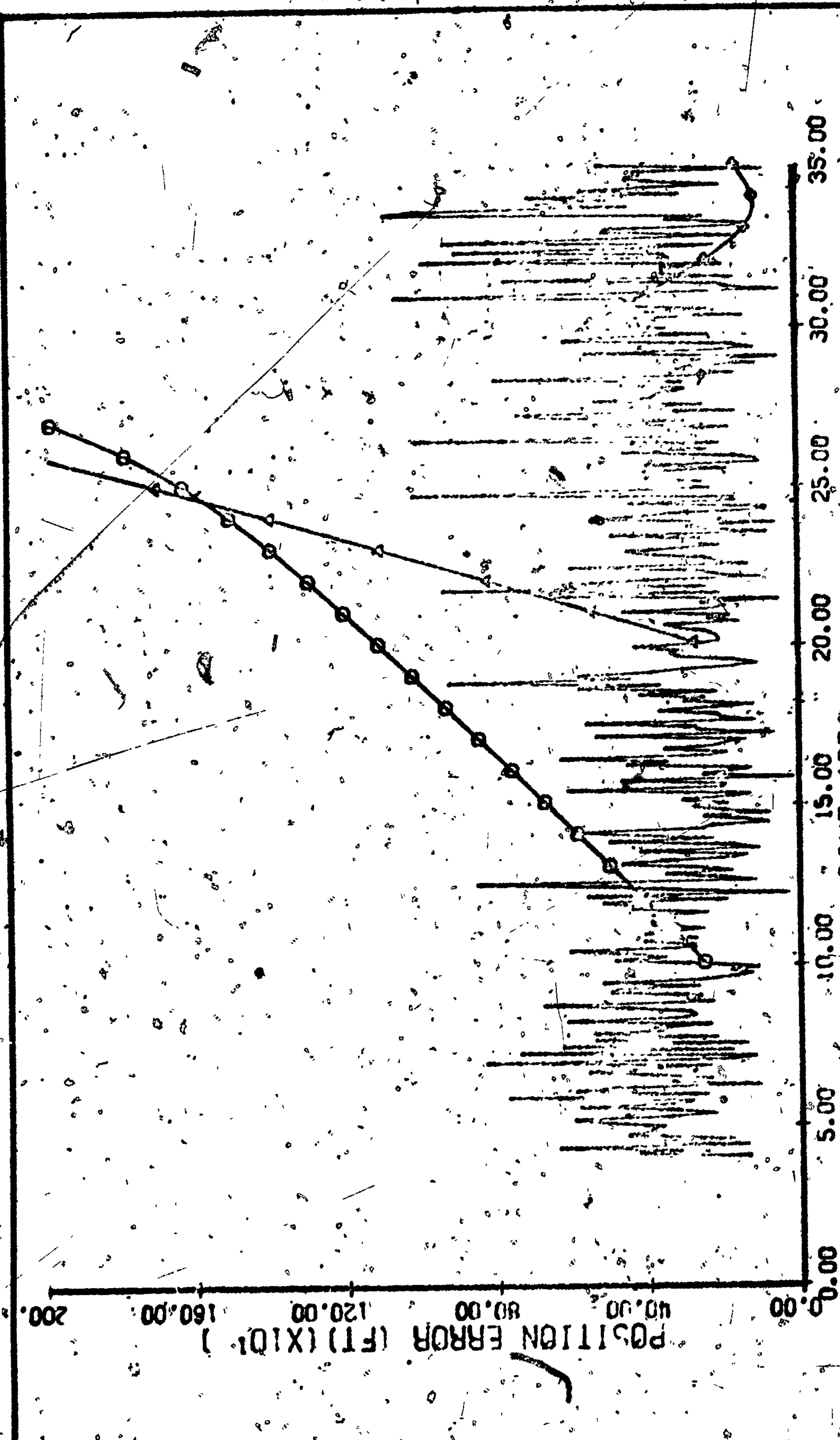


Figure 14 Position and Prediction Errors

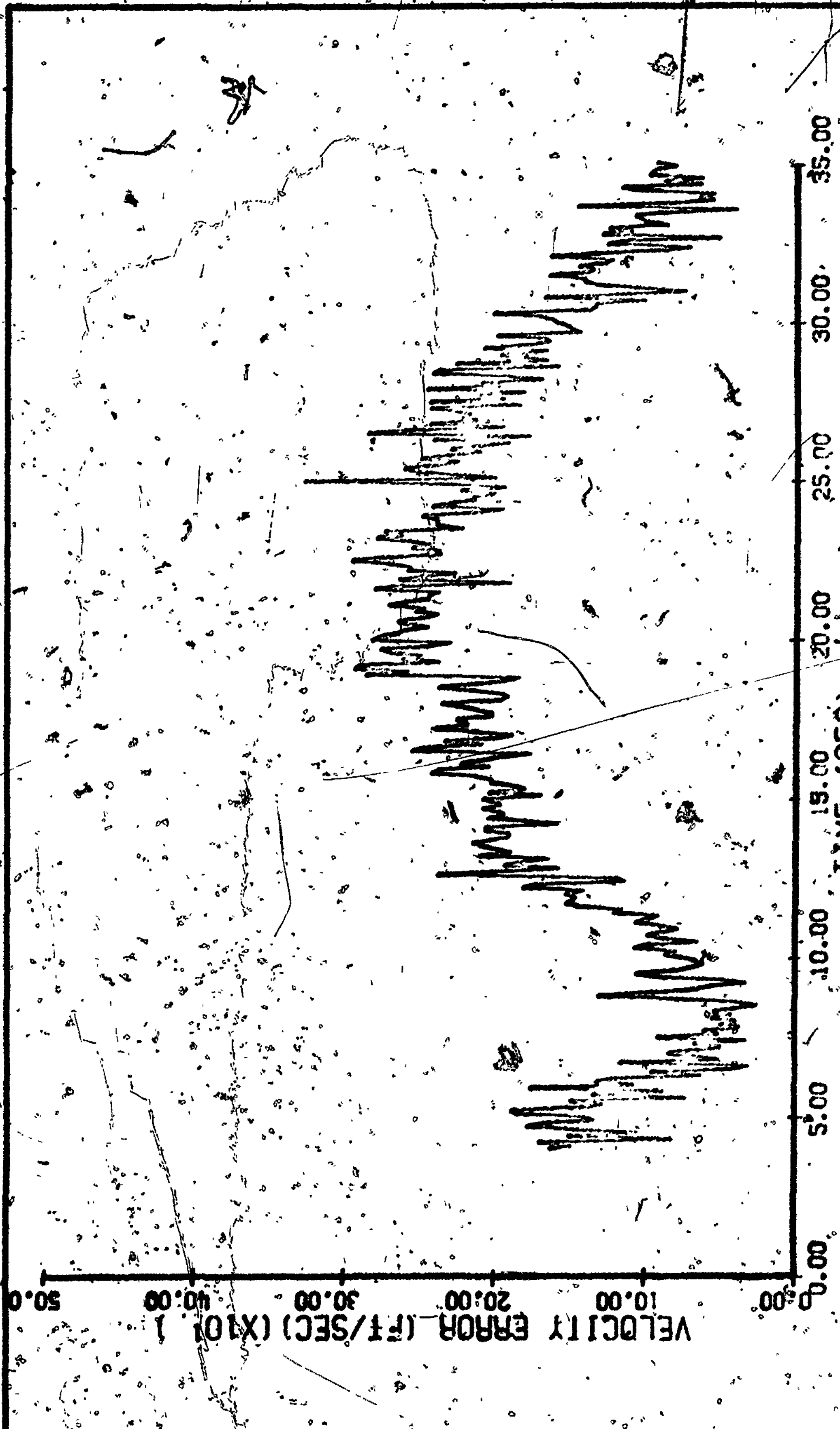


Figure 15 Velocity Error

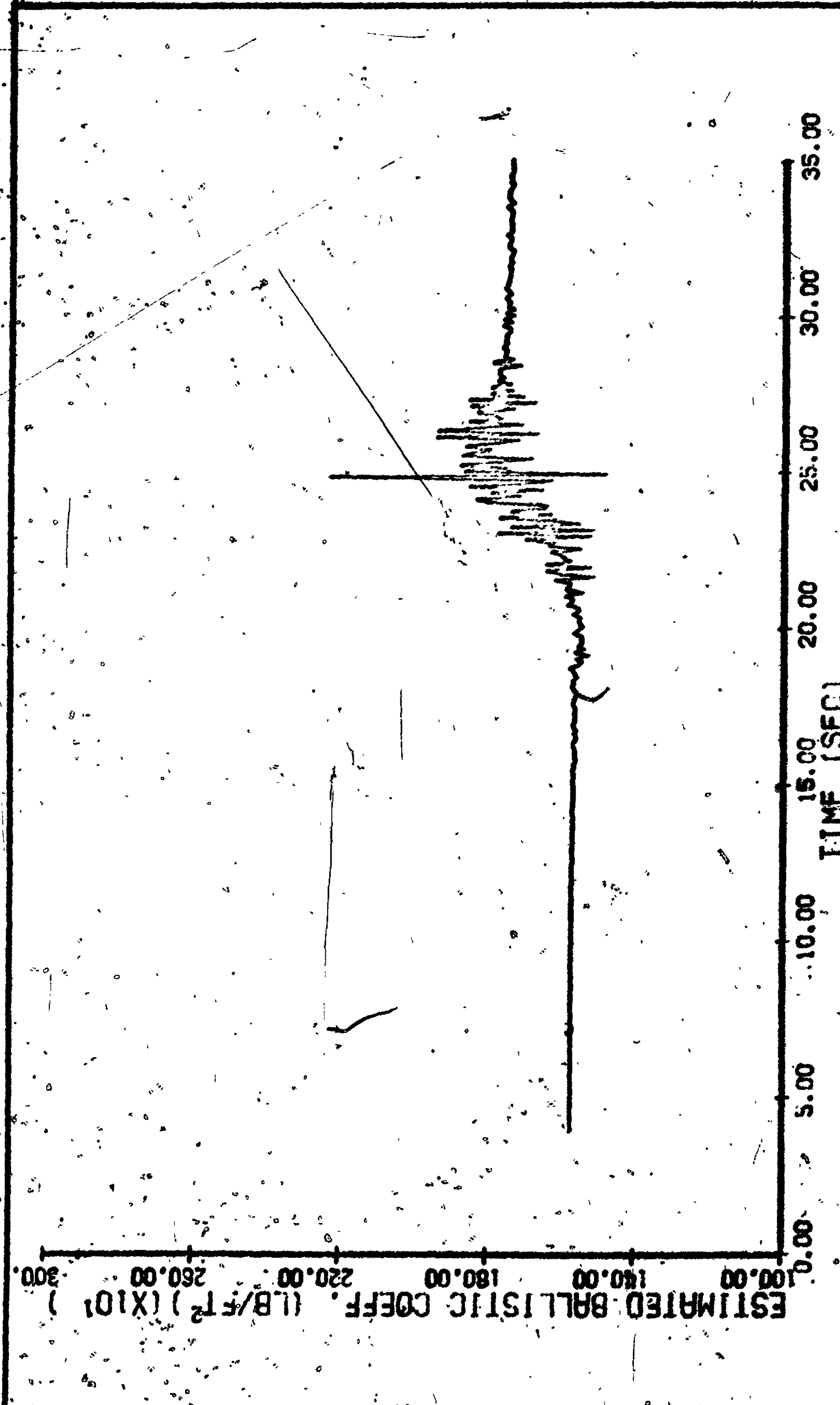


Figure 16 Estimated Ballistic Coefficient

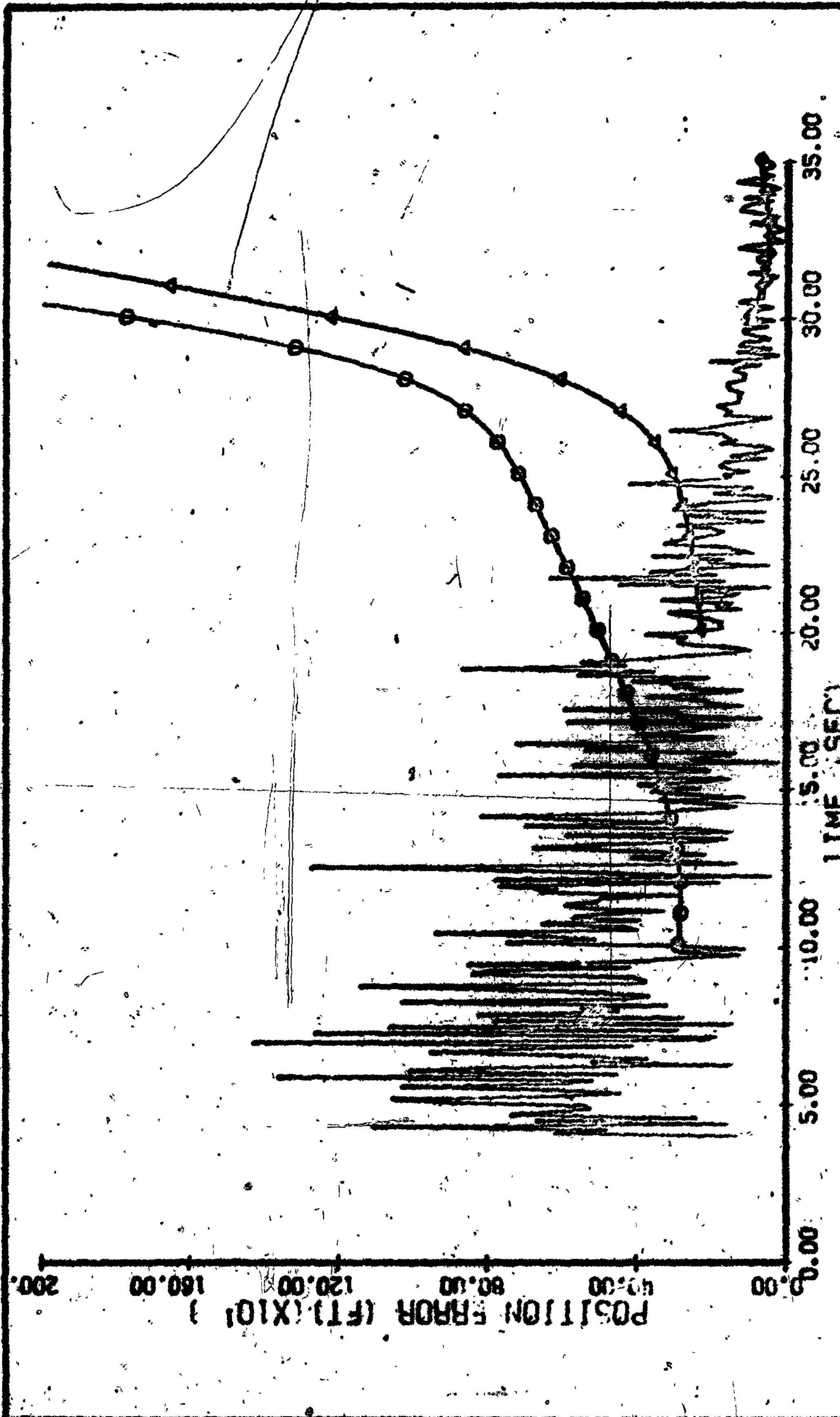


Figure 17 Position and Prediction Errors

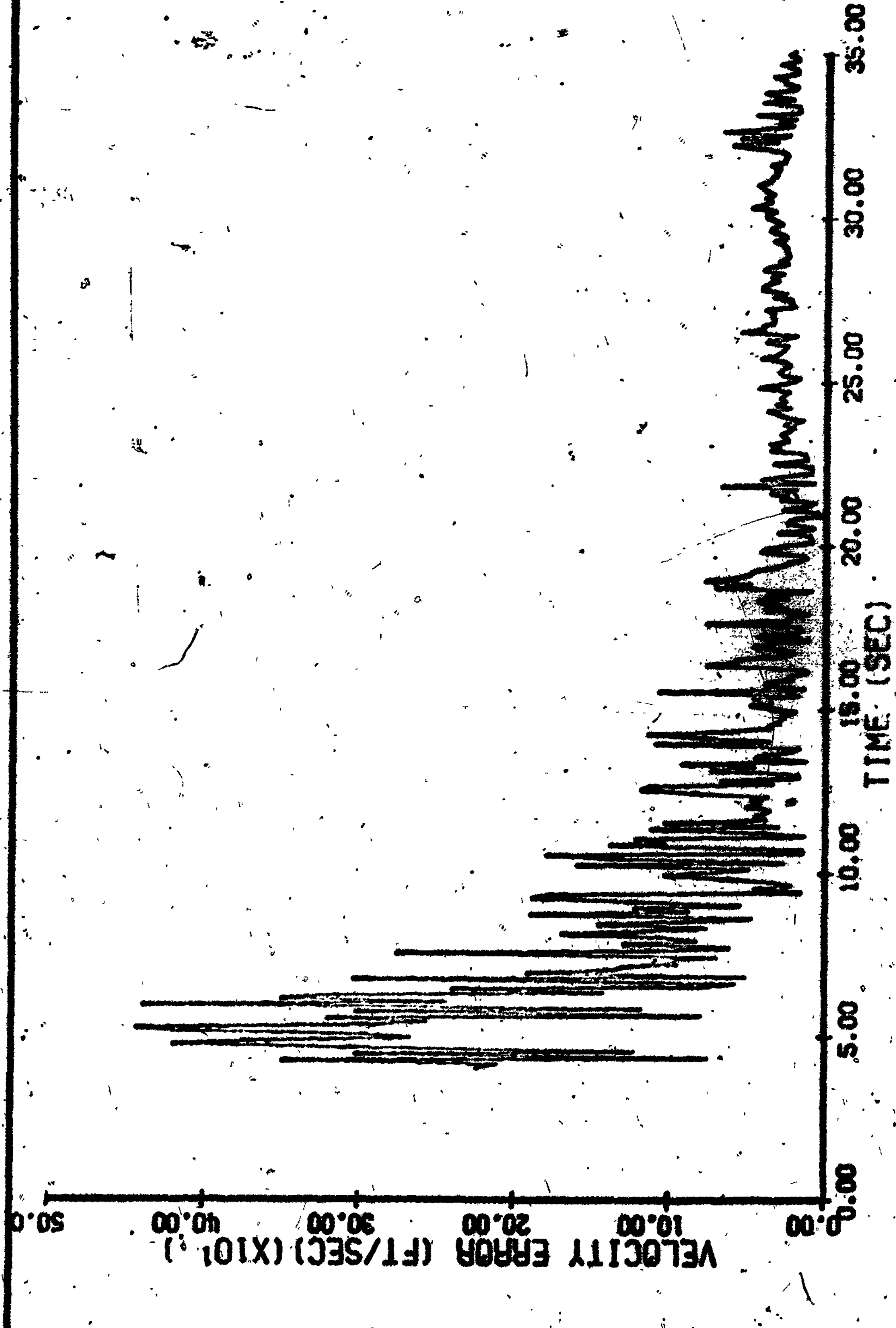


Figure 18 Velocity Error

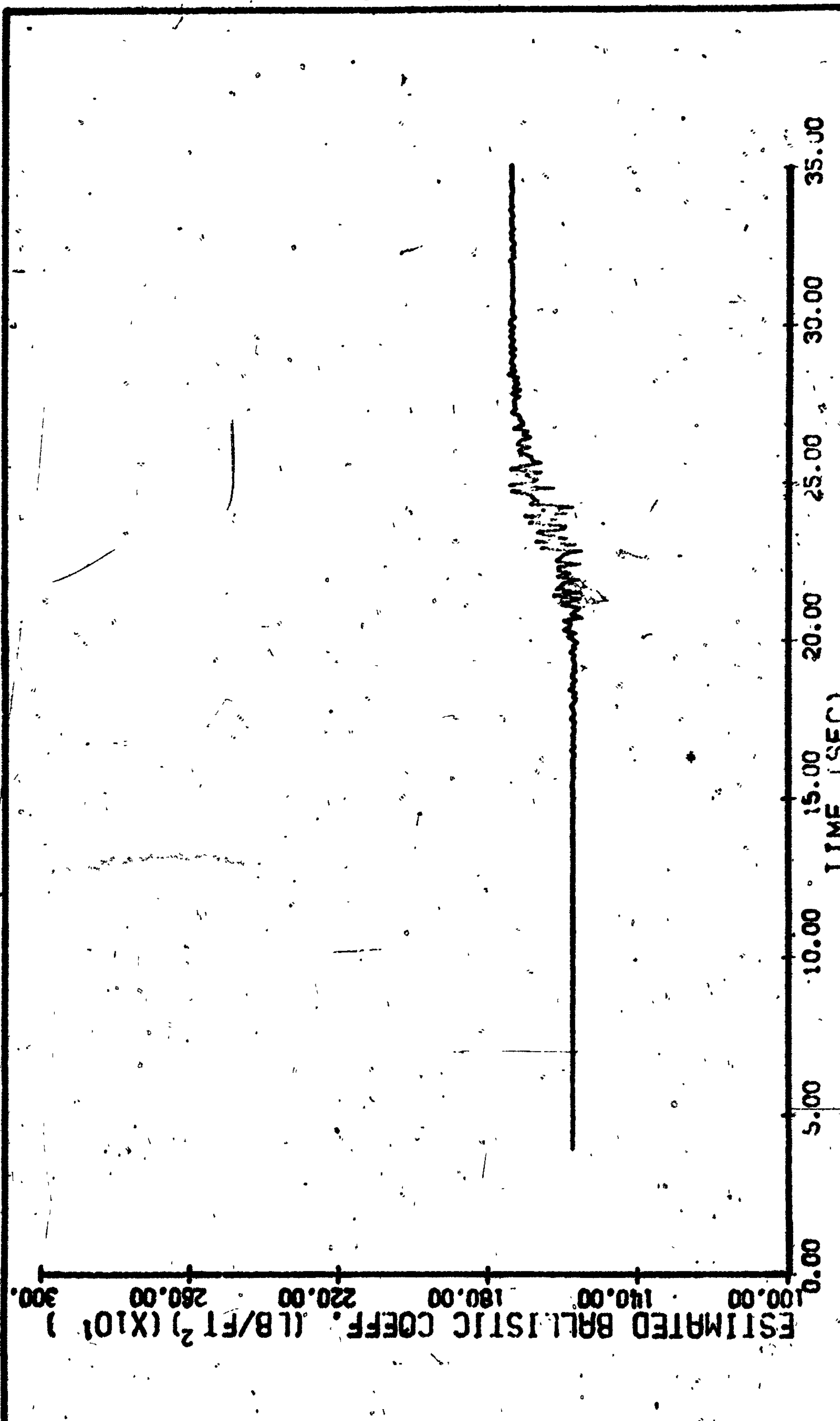


Figure 19 Estimated Ballistic Coefficient

Bibliography

1. Kalman, R.E., "A New Approach to Linear Filtering and Prediction Problems", A.S.M.E. Transaction Journal of Basic Engineering, 82D: 35-45, 1960.

2. Kalman, R.E., "New Results in Linear Filtering and Prediction Theory", A.S.M.E. Transaction Journal of Basic Engineering, 83D: 95-108, 1961.

3. Kalman, R.E., "New Methods and Results in Linear Prediction and Filtering", R.I.A.S., TR61-1; also published as "New Methods in Wiener Filtering", Proceedings of the First Symposium on Engineering Applications of Random Function Theory and Probability, John Wiley and Sons, Inc., 270-388, New York, 1963.

4. Schmidt, S.F., "State Space Techniques Applied to the Design of a Space Navigation System", Proceedings of the Joint Automatic Control Conference, 1962.

5. Sherman, S., "Study to Investigate the Stability in Orbit Determination", Technical Summary Report, NASA Contractor Report, NASA-CR-64942, April 1965.

6. Sorenson, H.W., "Kalman Filtering Techniques", Advances in Control Systems, Volume 3. Edited by C.T. Leondes, Academic Press, Inc., New York, 1966.

7. --------, "Modularized Six-Degree-of-Freedom (MOD6DF) Computer Program," Litton Systems, Inc., March 1966.

8. --------, "Digital Computer Simulation of Interceptor-Missiles", IIT Reasearch Institute, Volume I, AL-TDR-64-12. Jan 1964.

Appendix A

Computer Listing

The following IBM 360 Scientific Subroutines were also used:

Subroutine MPRD
LOC
MTRA
MCPY

The following 7094 Fortran IV Function was also used:
DET - Determinant Evaluating Function

```
$IBFTC EXEC.
      COMMON C(999)
      EQUIVALENCE       (C(001),T     ),(C(002),TF     ),
     1                  (C(006),STEP  )
    1 CALL ZERO
    2 CALL INPUT
      LSTEP=STEP
    3 CALL INITAL
    4 CALL OUPTI
    5 CALL ACTION
      CALL OUTPUT
      IF(T.LT.TF) GO TO 5
      CALL RESET
      GO TO (1,2,3,4,5,6),LSTEP
    6 STOP
      END
$IBFTC ZERO.   DECK
C
C     SUBROUTINE ZERO SETS INDICATORS AND CONSTANTS
C
      SUBROUTINE ZERO
      COMMON C(999)
      REAL MU
      EQUIVALENCE       (C(490),NORNDM),(C(499),NOLIST),(C(500),NOOUT ),
     1                  (C(011),RE    ),(C(012),MU    ),(C(006),STEP  ),
     2                  (C(013),WIE   ),(C(014),WIE2  ),
     3                  (C(222),F14   ),(C(230),F25   ),(C(238),F36   )
      DO 1 I=1,999
    1 C(I)=-0.0
      NORNDM=0
      NOOUT=0
      STEP=2.0
      RE=20926428.0
      MU=1.40775 E16
      WIE=7.2722E-5
      WIE2=WIE*WIE
      F14=1.0
      F25=1.0
      F36=1.0
      RETURN
      END
$IBFTC INITL.
      SUBROUTINE INITAL
      CALL ATMOSI
      CALL TRAJMI
      CALL PLANEI
      CALL NOISEI
      CALL MATCH
      CALL KALMAI
      CALL PREDTI
      CALL COMPAI
      RETURN
      END
$IBFTC ACTIO.
      SUBROUTINE ACTION
      COMMON C(999)
      DIMENSION PTIME(4)
      INTEGER PKOUNT
      EQUIVALENCE       (C(001),T     ),(C(008),TTSKF ),
     1                  (C(016),PKOUNT),(C(017),PTIME )
    1 CALL MISSLE
      CALL PLANE
      CALL NOISE
      CALL RADAR
    3 CALL KALMAN
      CALL COMPAR
      IF(PTIME(PKOUNT).LE.0.0) RETURN
      IF(T.LT.PTIME(PKOUNT)) RETURN
      CALL PREDIC
      RETURN
      END
```

```
$IBFTC MISSL.  DECK
C
C     SUBROUTINE MISSLE GENERATES THE REFERENCE TRAJECTORY IN
C     EARTH AND TANGENT PLANE COORDINATES
C
      SUBROUTINE MISSLE
      COMMON C(999)
      EQUIVALENCE     (C(101),XEM    ),(C(102),YEM    ),(C(103),ZEM    ),
     1                (C(104),VXEM   ),(C(105),VYEM   ),(C(106),VZEM   ),
     2                (C(120),XTM    ),(C(121),YTM    ),(C(122),ZTM    ),
     3                (C(123),VXTM   ),(C(124),VYTM   ),(C(125),VZTM   ),
     4                (C(031),CET11  ),(C(034),CET12  ),(C(037),CET13  ),
     5                (C(032),CET21  ),(C(035),CET22  ),(C(038),CET23  ),
     6                (C(033),CET31  ),(C(036),CET32  ),(C(039),CET33  ),
     7(C(011),RE     ),(C(108),H      ),(C(109),V      ),(C(110),Q      )
      CALL TRAJM
      H=SQRT(XEM*XEM+YEM*YEM+ZEM*ZEM)-RE
      V=SQRT(VXEM*VXEM+VYEM*VYEM+VZEM*VZEM)
      CALL ATMOS(H,RHO,GAMA)
      Q=0.5*RHO*V*V
      XTM=CET11*XEM+CET12*YEM+CET13*ZEM
      YTM=CET21*XEM+CET22*YEM+CET23*ZEM
      ZTM=CET31*XEM+CET32*YEM+CET33*ZEM
      VXTM=CET11*VXEM+CET12*VYEM+CET13*VZEM
      VYTM=CET21*VXEM+CET22*VYEM+CET23*VZEM
      VZTM=CET31*VXEM+CET32*VYEM+CET33*VZEM
      RETURN
      END
```

```
$IBFTC TRAJM.  DECK
C
C     INTEGRATION ROUTINE FOR THE REFERENCE TRAJECTORY
C     ADAMS-BASHFORTH  -  ADAMS-MOULTON  PREDICTOR-CORRECTOR WITH RUNGE-KUTTA
C
      SUBROUTINE TRAJMI
      COMMON C(999)
      DOUBLE PRECISION W
      DIMENSION D(6,5),W(6,5),Y(6),YD(6)
      EQUIVALENCE      (C(003),H      ),(C(001),X      ),(C(101),Y      ),
     1                 (C(111),YD     )
      DATA M/6/
      K=0
      K2=0
      DO 10 I=1,M
   10 W(I,1)=DBLE(Y(I))
      CALL DERT
      DO 1 I=1,6
    1 D(I,5)=YD(I)
      RETURN
      ENTRY TRAJM
   40 XC=X
      IF (K.NE.0) IF (K-2) 50,50,110
      XP=XC
      DO 45 I=1,M
   45 W(I,5)=W(I,1)
   50 K1=4-K
      DO 70 I=1,M
      DO 60 J=K1,4
   60 D(I,J)=D(I,J+1)
      W(I,2)=H*D(I,4)
      W(I,1)=W(I,1)+.5D0*W(I,2)
   70 Y(I)=SNGL(W(I,1))
      X=XC+.5*H
      CALL DERT
      DO 2 I=1,6
    2 D(I,5)=YD(I)
      DO 80 I=1,M
      W(I,3)=H*D(I,5)
      W(I,1)=W(I,1)+.5D0*(W(I,3)-W(I,2))
   80 Y(I)=SNGL(W(I,1))
      CALL DERT
      DO 3 I=1,6
    3 D(I,5)=YD(I)
      DO 90 I=1,M
      W(I,4)=H*D(I,5)
      W(I,1)=W(I,1)+W(I,4)-.5D0*W(I,3)
   90 Y(I)=SNGL(W(I,1))
      X=XC+H
      CALL DERT
      DO 4 I=1,6
    4 D(I,5)=YD(I)
      DO 100 I=1,M
      W(I,1)=W(I,1)-W(I,4)+.16666666666666667*(W(I,2)+2.D0*(W(I,3)+W(I,4
     1))+H*D(I,5))
  100 Y(I)=SNGL(W(I,1))
      K=K+1
      K1=K
      CALL DERT
      DO 5 I=1,6
    5 D(I,5)=YD(I)
      RETURN
  110 DO 130 I=1,M
      W(I,2)=W(I,1)
      DO 120 J=1,4
  120 D(I,J)=D(I,J+1)
      W(I,3)=W(I,2)+.41666666666666667D-1*H*(55.*D(I,4)-59.*D(I,3)+37.*D
     1(I,2)-9.*D(I,1))
  130 Y(I)=SNGL(W(I,3))
      X=XC+H
      CALL DERT
      DO 6 I=1,6
    6 D(I,5)=YD(I)
```

```
      DO 140 I=1,M
      W(I,1)=W(I,2)+.41666666666666667D-1*H*(9.*D(I,5)+19.*D(I,4)-5.*D(I
     1,3)+D(I,2))
  140 Y(I)=SNGL(W(I,1))
      CALL DERP
      DO 7 I=1,6
    7 D(I,5)=YD(I)
      RETURN
      END
```

```
$IBFTC DERP.   DECK
C
C      SUBROUTINE DERP PROVIDES THE DERIVATIVE LIST FOR THE INTEGRATION
C     ROUTINE FOR THE PREDICTION SUBROUTINE - TANGENT PLANE
C
      SUBROUTINE DERP
      COMMON C(999)
      REAL MU
      EQUIVALENCE      (C(987),X      ),(C(988),Y      ),(C(989),Z      ),
     1                 (C(990),VX     ),(C(991),VY     ),(C(992),VZ     ),
     2                 (C(021),WX     ),(C(022),WY     ),(C(023),WZ     ),
     3(C(993),ALPHA ),(C(012),MU     ),(C(011),RE     ),(C(014),WIE2   ),
     4                 (C(994),XD     ),(C(995),YD     ),(C(996),ZD     ),
     5                 (C(997),VXD    ),(C(998),VYD    ),(C(999),VZD    )
      R=SQRT(X*X+Y*Y+Z*Z)
      V=SQRT(VX*VX+VY*VY+VZ*VZ)
      G=MU/(R**3)
      H=R-RE
      CALL ATMOS(H,RHO,GAMA)
      D=0.5*RHO    *V*ALPHA
      SUM=WX*X+WY*Y+WZ*Z
      XD=VX
      YD=VY
      ZD=VZ
      VXD=-G*X-D*VX-2.0*(WY*VZ-WZ*VY)-WX*SUM+X*WIE2
      VYD=-G*Y-D*VY-2.0*(WZ*VX-WX*VZ)-WY*SUM+Y*WIE2
      VZD=-G*Z-D*VZ-2.0*(WX*VY-WY*VX)-WZ*SUM+Z*WIE2
      RETURN
      END
```

```
$IBFTC PLANI.  DECK
C
C     SUBROUTINE PLANE - AIRCRAFT MODEL - GENERATES AIRCRAFT POSITION AND
C     AIRCRAFT-TO-EARTH DIRECTION COSINES
C
      SUBROUTINE PLANEI
      COMMON C(999)
      REAL LAT,LONG,LATR,LONGR
      EQUIVALENCE     (C(126),LAT   ),(C(127),LONG  ),(C(128),HP    ),
     1                (C(129),HEAD  ),(C(130),VP    ),(C(137),GAMMA ),
     2                (C(131),XEP   ),(C(132),YEP   ),(C(133),ZEP   ),
     3                (C(134),VXEP  ),(C(135),VYEP  ),(C(136),VZEP  ),
     4                (C(041),CAE11 ),(C(044),CAE12 ),(C(047),CAE13 ),
     5                (C(042),CAE21 ),(C(045),CAE22 ),(C(048),CAE23 ),
     6                (C(043),CAE31 ),(C(046),CAE32 ),(C(049),CAE33 ),
     7                (C(011),RE    ),(C(001),T     )
      DATA CDTR/1.7453293E-2/
      LATR=LAT*CDTR
      LONGR=LONG*CDTR
      HEADR=HEAD*CDTR
      SLONG=SIN(LONGR)
      CLONG=COS(LONGR)
      SLAT=SIN(LATR)
      CLAT=COS(LATR)
      SHEAD=SIN(HEADR)
      CHEAD=COS(HEADR)
C
C     CALCULATE INITIAL AIRCRAFT-TO-EARTH DIRECTION COSINES
C
      CAE11=-SHEAD*SLONG-SLAT*CHEAD*CLONG
      CAE21=SHEAD*CLONG-SLAT*CHEAD*SLONG
      CAE31=CLAT*CHEAD
      CAE12=CHEAD*SLONG-SLAT*SHEAD*CLONG
      CAE22=-CHEAD*CLONG-SLAT*SHEAD*SLONG
      CAE32=CLAT*SHEAD
      CAE13=CLAT*CLONG
      CAE23=CLAT*SLONG
      CAE33=SLAT
      R=RE+HP
      XO=CAE13*R
      YO=CAE23*R
      ZO=CAE33*R
      XEP=XO
      YEP=YO
      ZEP=ZO
      VXEP=CAE11*VP
      VYEP=CAE21*VP
      VZEP=CAE31*VP
      RETURN
      ENTRY PLANE
C
C     CALCULATE NEW AIRCRAFT POSITION
C
      IF (VP.EQ.0.0)  RETURN
      XEP=XO+VXEP*T
      YEP=YO+VYEP*T
      ZEP=ZO+VZEP*T
      P2=XEP*XEP+YEP*YEP
      P=SQRT(P2)
      R=SQRT(P2+ZEP*ZEP)
      HP=R-RE
C
C     UPDATE AIRCRAFT-TO-EARTH DIRECTION COSINES
C
      E11=-YEP/P
      E21=XEP/P
      E31=0.0
      E13=XEP/R
      E23=YEP/R
      E33=ZEP/R
      E12=-E21*E33
      E22=E11*E33
      E32=P/R
```

```
VE=E11*VXEP+E21*VYEP+E31*VZEP
VN=E12*VXEP+E22*VYEP+E32*VZEP
VR=E13*VXEP+E23*VYEP+E33*VZEP
VH=SQRT(VE*VE+VN*VN)
SHEAD=VE/VH
CHEAD=VN/VH
GAMMA=ATAN2(VR,VH)
CAE11=E11*SHEAD+E12*CHEAD
CAE21=E21*SHEAD+E22*CHEAD
CAE31=E31*SHEAD+E32*CHEAD
CAE12=-E11*CHEAD+E12*SHEAD
CAE22=-E21*CHEAD+E22*SHEAD
CAE32=-E31*CHEAD+E32*SHEAD
CAE13=E13
CAE23=E23
CAE33=E33
RETURN
END
```

```
$IBFTC RADAR.  DECK
C
C     SUBROUTINE RADAR GENERATES RADAR MEASUREMENT DATA
C
      SUBROUTINE RADAR
      COMMON C(999)
      DATA CRTD/57.295779/
      EQUIVALENCE       (C(101),XEM   ),(C(102),YEM   ),(C(103),ZEM   ),
     1                  (C(104),VXEM  ),(C(105),VYEM  ),(C(106),VZEM  ),
     2                  (C(131),XEP   ),(C(132),YEP   ),(C(133),ZEP   ),
     3                  (C(134),VXEP  ),(C(135),VYEP  ),(C(136),VZEP  ),
     4                  (C(041),CAE11 ),(C(044),CAE12 ),(C(047),CAE13 ),
     5                  (C(042),CAE21 ),(C(045),CAE22 ),(C(048),CAE23 ),
     6                  (C(043),CAE31 ),(C(046),CAE32 ),(C(049),CAE33 ),
     7(C(067),EPSAZ ),(C(077),EPSEL ),(C(087),EPSRA ),(C(097),EPSRR ),
     8(C(070),AZ    ),(C(080),EL    ),(C(090),RA    ),(C(100),RR    ),
     9(C(040),AZD   ),(C(050),ELD   )
      X=XEM-XEP
      Y=YEM-YEP
      Z=ZEM-ZEP
      VX=VXEM-VXEP
      VY=VYEM-VYEP
      VZ=VZEM-VZEP
      XA=CAE11*X+CAE21*Y+CAE31*Z
      YA=CAE12*X+CAE22*Y+CAE32*Z
      ZA=CAE13*X+CAE23*Y+CAE33*Z
      AZ=ATAN2(-YA,XA)+EPSAZ
      XYR=SQRT(XA*XA+YA*YA)
      EL=ATAN2(ZA,XYR)+EPSEL
      R=SQRT(X*X+Y*Y+Z*Z)
      RA=R+EPSRA
      RR=((X*VX+Y*VY+Z*VZ)/R)+EPSRR
      AZD=AZ*CRTD
      ELD=EL*CRTD
      RETURN
      END
```

```
$IBFTC IGUES.  DECK
      SUBROUTINE IGUESI
      COMMON C(999)
      DIMENSION A(6),AX(3),AY(3),AZ(3),BX(3),BY(3),BZ(3)
      EQUIVALENCE       (C(031),CET11 ),(C(034),CET12 ),(C(037),CET13 ),
     1                  (C(032),CET21 ),(C(035),CET22 ),(C(038),CET23 ),
     2                  (C(033),CET31 ),(C(036),CET32 ),(C(039),CET33 ),
     3(C(013),WIE   ),(C(021),WX    ),(C(022),WY    ),(C(023),WZ    ),
     4                  (C(041),CAE11 ),(C(044),CAE12 ),(C(047),CAE13 ),
     5                  (C(042),CAE21 ),(C(045),CAE22 ),(C(048),CAE23 ),
     6                  (C(043),CAE31 ),(C(046),CAE32 ),(C(049),CAE33 ),
     7                  (C(141),XTP   ),(C(142),YTP   ),(C(143),ZTP   ),
     8                  (C(144),VXTP  ),(C(145),VYTP  ),(C(146),VZTP  ),
     9(C(009),TK    ),(C(001),T     ),(C(008),TTSKF ),
     1(C(070),AZ    ),(C(080),EL    ),(C(090),RA    ),(C(011),RE    ),
     2                  (C(101),XEM   ),(C(102),YEM   ),(C(103),ZEM   ),
     3                  (C(104),VXEM  ),(C(105),VYEM  ),(C(106),VZEM  ),
     4(C(107),BETA  ),(C(140),EBETA ),(C(169),SIGEL ),(C(350),SIGB  ),
     5(C(401),PP11  ),(C(403),PP22  ),(C(406),PP33  ),(C(410),PP44  ),
     6(C(415),PP55  ),(C(421),PP66  ),(C(428),PP77  ),
     7(C(128),HP    ),(C(120),XTM   ),(C(121),YTM   ),(C(122),ZTM   ),
     8                  (C(123),VXTM  ),(C(124),VYTM  ),(C(125),VZTM  )
C
C     INITIALIZE THE ROUTINE
C
      TO=T
      DO 1 I=1,3
      BX(I)=0.0
      BY(I)=0.0
    1 BZ(I)=0.0
      DO 2 I=1,6
    2 A(I)=0.0
      XOA=XEM
      YOA=YEM
      ZOA=ZEM
      RETURN
C
      ENTRY IGUESS
C
C     COMPUTE POSITION IN EARTH COORDINATES FROM RADAR OBSERVATIONS
C
      COSEL=COS(EL)
      XA=RA*COSEL*COS(AZ)
      YA=-RA*COSEL*SIN(AZ)
      ZA=RA*SIN(EL)+RE+HP
      X=CAE11*XA+CAE12*YA+CAE13*ZA
      Y=CAE21*XA+CAE22*YA+CAE23*ZA
      Z=CAE31*XA+CAE32*YA+CAE33*ZA
C
C     LOAD MATRICES FOR LEAST SQUARES FIT
C
      T2=T*T
      T3=T2*T
      A(1)=A(1)+1.0
      A(2)=A(2)+T
      A(3)=A(3)+T2
      A(5)=A(5)+T3
      A(6)=A(6)+T3*T
      BX(1)=BX(1)+X
      BX(2)=BX(2)+X*T
      BX(3)=BX(3)+X*T2
      BY(1)=BY(1)+Y
      BY(2)=BY(2)+Y*T
      BY(3)=BY(3)+Y*T2
      BZ(1)=BZ(1)+Z
      BZ(2)=BZ(2)+Z*T
      BZ(3)=BZ(3)+Z*T2
      IF(T.LT.(TTSKF-0.0005)) RETURN
      A(4)=A(3)
C
C     COMPUTE COEFFICIENTS OF POLYNOMIALS FOR LEAST SQUARES FIT
C
      CALL SINV(A,3,1.0E-5,IER)
```

```
      CALL MPRD(A,BX,AX,3,3,1,0,1)
      CALL MPRD(A,BY,AY,3,3,1,0,1)
      CALL MPRD(A,BZ,AZ,3,3,1,0,1)
C
C     COMPUTE ESTIMATED POSITION AND VELOCITY AT TIME T
C
      X1=AX(1)+AX(2)*T+AX(3)*T2
      Y1=AY(1)+AY(2)*T+AY(3)*T2
      Z1=AZ(1)+AZ(2)*T+AZ(3)*T2
      VX1=AX(2)+2.0*AX(3)*T
      VY1=AY(2)+2.0*AY(3)*T
      VZ1=AZ(2)+2.0*AZ(3)*T
C
C     COMPUTE ESTIMATED POSITION AT TIME TO
C
      XO=((AX(3)*TO)+AX(2))*TO+AX(1)
      YO=((AY(3)*TO)+AY(2))*TO+AY(1)
      ZO=((AZ(3)*TO)+AZ(2))*TO+AZ(1)
C
C     ESTABLISH TANGENT PLANE COORDINATE SYSTEM AND COMPUTE DIRECTION
C     COSINES FOR EARTH-TO-TANGENT PLANE COORDINATE TRANSFORMATION
C
      C1=YO*Z1-Y1*ZO
      C2=ZO*X1-XO*Z1
      C3=XO*Y1-X1*YO
      D=SQRT(C1*C1+C2*C2+C3*C3)
      CET21=C1/D
      CET22=C2/D
      CET23=C3/D
      C1=CET22*ZO-YO*CET23
      C2=CET23*XO-ZO*CET21
      C3=CET21*YO-XO*CET22
      D=SQRT(C1*C1+C2*C2+C3*C3)
      CET11=C1/D
      CET12=C2/D
      CET13=C3/D
      D=SQRT(XO*XO+YO*YO+ZO*ZO)
      CET31=XO/D
      CET32=YO/D
      CET33=ZO/D
C
C     COMPUTE COMPONENTS OF EARTH ROTATION IN TANGENT PLANE
C
      WX=CET13*WIE
      WY=CET23*WIE
      WZ=CET33*WIE
C
C     COMPUTE INITIAL ESTIMATE OF POSITION AND VELOCITY FOR KALMAN FILTER
C
      XTP=CET11*X1+CET12*Y1+CET13*Z1
      YTP=CET21*X1+CET22*Y1+CET23*Z1
      ZTP=CET31*X1+CET32*Y1+CET33*Z1
      VXTP=CET11*VX1+CET12*VY1+CET13*VZ1
      VYTP=CET21*VX1+CET22*VY1+CET23*VZ1
      VZTP=CET31*VX1+CET32*VY1+CET33*VZ1
C
C     COMPUTE DIFFERENCE BETWEEN ACTUAL AND ESTIMATED VALUES
C     OF POSITION AND VELOCITY
C
      DXO=XOA-XO
      DYO=YOA-YO
      DZO=ZOA-ZO
      DX1=XEM-X1
      DY1=YEM-Y1
      DZ1=ZEM-Z1
      DVX1=VXEM-VX1
      DVY1=VYEM-VY1
      DVZ1=VZEM-VZ1
      DBETA=BETA-EBETA
C
C     COMPUTE INITIAL VALUES FOR STATE COVARIANCE MATRIX
C
      SIGR=SIGEL*RA
```

```
      SIGR2=SIGR*SIGR
      SIGV=SIGR/T
      SIGV2=SIGV*SIGV
      IF(SIGB.EQ.0.0) SIGB=100.0
      PP11=SIGR2
      PP22=SIGR2
      PP33=SIGR2
      PP44=SIGV2
      PP55=SIGV2
      PP66=SIGV2
      PP77=1.0/(SIGB*SIGB)
C
C     OUTPUT CONDITIONS FOR START OF KALMAN FILTERING
C
      WRITE(6,600) AX,AY,AZ,XOA,XO,DXO,YOA,YO,DYO,ZOA,ZO,DZO,T,XEM,X1,
     1DX1,SIGR,YEM,Y1,DY1,SIGR,ZEM,Z1,DZ1,SIGR,VXEM,VX1,DVX1,SIGV ,
     2VYEM,VY1,DVY1,SIGV ,VZEM,VZ1,DVZ1,SIGV ,BETA,EBETA,DBETA,SIGB
  600 FORMAT(18H1LEAST SQUARES FIT/1HA,62X,1H2/7H X  =  1PE14.7,5H  +  ,
     1E14.7,7H T  +  ,E14.7,2H T/1HA,62X,1H2/7H Y  =  ,E14.7,5H  +  ,
     2E14.7,7H T  +  ,E14.7,2H T/1HA,62X,1H2/7H Z  =  ,E14.7,5H  +  ,
     3E14.7,7H T  +  ,E14.7,2H T///1HA,14X,6HACTUAL,11X,9HESTIMATED,
     46X,10HDIFFERENCE,10X,5HSIGMA/17H0TIME = 0 SECONDS/7HAX0   =,3E18.7
     5/7H0Y0   =,3E18.7/7H0Z0   =,3E18.7/7HATIME =,0PF5.2,8H SECONDS/
     67HAX1   =,1P4E18.7/7H0Y1   =,4E18.7/7H0Z1   =,4E18.7/7HAVX1  =,
     74E18.7/7H0VY1  =,4E18.7/7H0V   =,4E18.7/7HABETA =,4E18.7)
C
      XTM=CET11*XEM+CET12*YEM+CET13*ZEM
      YTM=CET21*XEM+CET22*YEM+CET23*ZEM
      ZTM=CET31*XEM+CET32*YEM+CET33*ZEM
      VXTM=CET11*VXEM+CET12*VYEM+CET13*VZEM
      VYTM=CET21*VXEM+CET22*VYEM+CET23*VZEM
      VZTM=CET31*VXEM+CET32*VYEM+CET33*VZEM
      CALL COMPAR
      CALL OUTPUT
      TK=T
      RETURN
      END
```

```
$IBFTC KALM.
C       X        (7X1)     STATE VECTOR  (TANGENT PLANE)
C       Z        (4X1)     VECTOR OF OBSERVABLES
C       K        (7X4)     FILTER GAIN MATRIX
C       R        (4X4)     MEASUREMENT NOISE COVARIANCE MATRIX
C       PE       (7X7)     FILTER ESTIMATION COVARIANCE MATRIX
C       PP       (7X7)     FILTER PREDICTION COVARIANCE MATRIX
C       PHI      (7X7)     STATE TRANSITION MATRIX
C       PHIT     (7X7)     TRANSPOSE OF STATE TRANSITION MATRIX
C       F        (7X7)     SYSTEM DESCRIPTION MATRIX
C       DXEST    (7X1)     VECTOR OF OPTIMAL ESTIMATION OF ERRORS IN STATES
C       CET      (3X3)     DIRECTION COSINES  (EARTH-TO-TARGRT)
C       CAE      (3X3)     DIRECTION COSINES  (AIRPLANE-TO EARTH)
C       CAT      (3X3)     DIRECTION COSINES  (AIRPLANE-TO TARGET)
C       PAD77    (7X7)     SCRATCH PAD
C       PAD74    (7X4)     SCRATCH PAD
      SUBROUTINE KALMAI
      COMMON C(999)
      INTEGER PGCNT
      REAL K(7,4),M44,M45,M46
      DIMENSION X(7),Z(4),CV(3),PE(7,7),PP(28),PHI(7,7),PHIT(7,7),R(7),
     1F(7,7),DXEST(7),CET(3,3),CAE(3,3),CAT(3,3),PAD77(7,7),PAD74(7,4),
     2PF74(7,4),PAD47(4,7),Q(10),A(7),W(3)
      EQUIVALENCE      (C(031),CET    ),(C(041),CAE    ),(C(051),CAT    ),
     1(C(070),AZ     ),(C(080),EL     ),(C(090),RA     ),(C(100),RR     ),
     2(C(141),X      ),(C(161),DXEST  ),(C(157),Z      ),(C(173),K      ),
     3(C(201),F      ),(C(251),PHI    ),(C(301),PE     ),(C(351),R      ),
     4(C(003),DT     ),(C(010),DT2    ),(C(011),RE     ),(C(130),VP     ),
     5(C(361),M44    ),(C(362),M45    ),(C(363),M46    ),(C(172),D      ),
     6(C(140),EBETA  ),(C(128),HP     ),(C(488),PGCNT  ),(C(401),PP     ),
     7(C(168),SIGAZ  ),(C(169),SIGEL  ),(C(170),SIGRA  ),(C(171),SIGRR  ),
     8(C(137),GAMMA  ),(C(015),EPS    ),(C(138),SEPR   ),(C(139),SEPV   ),
     9(C(148),H      ),(C(149),V      )
     1               ,(C(955),SEPR1  ),(C(956),SEPV1  )
      DT2=DT*DT/2.0
      SIGR2=SIGRR*SIGRR
      EPS2=EPS*EPS
      X(7)=1.0/EBETA
      RETURN
      ENTRY KALMAN
C
C     COMPUTE THE SYSTEM DESCRIPTION MATRIX - F
C
      CALL SDM
C
C     COMPUTE STATE TRANSITION MATRIX - PHI  AND  PHIT
C
      CALL MPRD(F,F,PAD77,7,7,0,0,7)
      DO 11 I=1,7
      DO 10 J=1,7
   10 PHI(I,J)=F(I,J)*DT+PAD77(I,J)*DT2
   11 PHI(I,I)=1.0+PHI(I,I)
      CALL MTRA(PHI,PHIT,7,7,0)
C
C     UPDATE FILTER ESTIMATION COVARIANCE MATRIX - PE
C
      CALL MPRD(PHI,PP,PAD77,7,7,0,1,7)
      CALL MPRD(PAD77,PHIT,PE,7,7,0,0,7)
      DO 15 I=1,7
   15 PE(I,I)=PE(I,I)+R(I)
      D=DET(PE,7)
      IF(D.EQ.0.0) WRITE(6,600)
  600 FORMAT(1HA,10X,10H**********,10X,35HSTATE COVARIANCE MATRIX IS SIN
     1GULAR   ,10X,10H**********)
C
C     UPDATE MEASUREMENT MATRIX - M
C
      SA=SIN(AZ)
      CA=COS(AZ)
      SE=SIN(EL)
      CE=COS(EL)
      CRA1=CE*CA
      CRA2=-CE*SA
```

```
      CRA3=SE
      CALL MPRD(CET,CAE,CAT,3,3,0,0,3)
      M44=CAT(1,1)*CRA1+CAT(1,2)*CRA2+CAT(1,3)*CRA3
      M45=CAT(2,1)*CRA1+CAT(2,2)*CRA2+CAT(2,3)*CRA3
      M46=CAT(3,1)*CRA1+CAT(3,2)*CRA2+CAT(3,3)*CRA3
C
C     CALCULATE THE MEASUREMENT NOISE COVARIANCE MATRIX - C
C
      RSIGA=RA*SIGAZ
      RSIGE=RA*SIGEL
      W(1)=CRA2*RSIGA-SE*CA*RSIGE+CRA1*SIGRA
      W(2)=-CRA1*RSIGA+SE*SA*RSIGE+CRA2*SIGRA
      W(3)=CE*RSIGE+SE*SIGRA
      CALL MPRD(CAT,W,CV,3,3,0,0,1)
C
C     COMPUTE FILTER GAIN MATRIX - K
C
      DO 20 I=1,7
   20 A(I)=M44*PE(4,I)+M45*PE(5,I)+M46*PE(6,I)
      Q(1)=PE(1,1)+CV(1)*CV(1)
      Q(2)=PE(1,2)+CV(1)*CV(2)
      Q(3)=PE(2,2)+CV(2)*CV(2)
      Q(4)=PE(1,3)+CV(1)*CV(3)
      Q(5)=PE(2,3)+CV(2)*CV(3)
      Q(6)=PE(3,3)+CV(3)*CV(3)
      Q(7)=A(1)
      Q(8)=A(2)
      Q(9)=A(3)
      Q(10)=M44*A(4)+M45*A(5)+M45*A(6)
      CALL SINV (Q,4,1.0E-05,IER)
      DO 22 I=1,7
      DO 21 J=1,3
   21 PAD74(I,J)=PE(I,J)
   22 PAD74(I,4)=A(I)
      CALL MPRD(PAD74,Q,PP74,7,4,0,1,4)
      CALL MTRA(PAD74,PAD47,7,4,0)
      CALL MPRD(PP74,PAD47,PAD77,7,4,0,0,7)
      PAD74(1,1)=PAD74(1,1)+EPS
      PAD74(2,2)=PAD74(2,2)+EPS
      PAD74(3,3)=PAD74(3,3)+EPS
      PAD74(4,4)=PAD74(4,4)+M44*EPS
      PAD74(5,4)=PAD74(5,4)+M45*EPS
      PAD74(6,4)=PAD74(6,4)+M46*EPS
      CALL MPRD(PAD74,Q,K,7,4,0,1,4)
C
C     UPDATE FILTER PREDICTION COVARIANCE MATRIX
C
      DO 30 I=1,6
   30 PP(I)=Q(I)*EPS2
      DO 31 I=7,10
      PP(I)=Q(I)*M44*EPS2
      J=I+4
      PP(J)=Q(I)*M45*EPS2
      KK=I+9
   31 PP(KK)=Q(I)*M46*EPS2
      PP(10)=PP(10)*M44
      PP(15)=PP(14)*M45
      PP(14)=PP(14)*M44
      PP(21)=PP(19)*M46
      PP(20)=PP(19)*M45
      PP(19)=PP(19)*M44
      DO 32 I=22,28
   32 PP(I)=0.0
      KK=1
      DO 33 J=1,7
      DO 33 I=1,J
      PP(KK)=PP(KK)+PE(I,J)-PAD77(I,J)
   33 KK=KK+1
      SEPR=SQRT(PP(1)*PP(1)+PP(3)*PP(3)+PP(6)*PP(6))
      SEPR1=SQRT(PP(1)+PP(3)+PP(6))
      SEPV=SQRT(PP(10)*PP(10)+PP(15)*PP(15)+PP(21)*PP(21))
      SEPV1=SQRT(PP(10)+PP(15)+PP(21))
C
```

```
C     INTEGRATE THE EQUATIONS OF MOTION
C
      CALL TRAJK
C
C     CALCULATE OPTIMUM ESTIMATE OF ERRORS IN STATES
C
      REHP=RE+HP
      Z(1)=X(1)-M44*RA-CAT(1,3)*REHP
      Z(2)=X(2)-M45*RA-CAT(2,3)*REHP
      Z(3)=X(3)-M46*RA-CAT(3,3)*REHP
      Z(4)=M44*X(4)+M45*X(5)+M46*X(6)-VP*(CRA1*COS(GAMMA)+SE*SIN(GAMMA))
     1-RR
      CALL MPRD(K,Z,DXEST,7,4,0,0,1)
C
C     UPDATE STATES
C
      DO 40 I=1,7
   40 X(I)=X(I)-DXEST(I)
      H=SQRT(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))-RE
      V=SQRT(X(4)*X(4)+X(5)*X(5)+X(6)*X(6))
      EDETA=1.0/X(7)
      RETURN
      END
```

```
$IBFTC SDM.    DECK
C
C      SUBROUTINE SDM COMPUTES THE SYSTEM DESCRIPTION MATRIX FOR KALMAN FILTER
C
       SUBROUTINE SDM
       COMMON C(999)
       REAL MU
       DIMENSION F(7,7)
       EQUIVALENCE     (C(011),RE    ),(C(012),MU    ),(C(201),F     ),
      1                (C(141),X     ),(C(142),Y     ),(C(143),Z     ),
      2                (C(144),VX    ),(C(145),VY    ),(C(146),VZ    ),
      3                (C(147),ALPHA ),(C(013),WIE   ),(C(014),WIE2  ),
      4                (C(021),WX    ),(C(022),WY    ),(C(023),WZ    )
      R=SQRT(X*X+Y*Y+Z*Z)
      V=SQRT(VX*VX+VY*VY+VZ*VZ)
      G=MU/R**3
      H=R-RE
      CALL ATMOS(H,RHO,PRHO)
      D=0.5*RHO*V*ALPHA
      T1=3.0*G/(R*R)
      T2=D*PRHO/(RHO*R)
      T3=D/(V*V)
      T4=-D/ALPHA
      TX=T1*X-T2*VX
      TX3=T3*VX
      F(4,1)=-G+TX*X-WX*WX+WIE2
      F(4,2)=   TX*Y-WX*WY
      F(4,3)=   TX*Z-WX*WZ
      F(4,4)=-D-TX3*VX
      F(4,5)=  -TX3*VY+2.0*WZ
      F(4,6)=  -TX3*VZ-2.0*WY
      F(4,7)=T4*VX
      TY=T1*Y-T2*VY
      TY3=T3*VY
      F(5,1)=   TY*X-WX*WY
      F(5,2)=-G+TY*Y-WY*WY+WIE2
      F(5,3)=   TY*Z-WY*WZ
      F(5,4)=  -TY3*VX-2.0*WZ
      F(5,5)=-D-TY3*VY
      F(5,6)=  -TY3*VZ+2.0*WX
      F(5,7)=T4*VY
      TZ=T1*Z-T2*VZ
      TZ3=T3*VZ
      F(6,1)=   TZ*X-WX*WZ
      F(6,2)=   TZ*Y-WY*WZ
      F(6,3)=-G+TZ*Z-WZ*WZ+WIE2
      F(6,4)=  -TZ3*VX+2.0*WY
      F(6,5)=  -TZ3*VY-2.0*WX
      F(6,6)=-D-TZ3*VZ
      F(6,7)=T4*VZ
      RETURN
      END
```

```
$IBFTC TRAJK.  DECK
C
C      INTEGRATION ROUTINE FOR KALMAN FILTER TRAJECTORY
C      DOUBLE PRECISION RUNGE-KUTTA
C
       SUBROUTINE TRAJK
       COMMON C(999)
       EQUIVALENCE        (C(009),T      ),(C(003),H      ),
      1                   (C(141),X      ),(C(151),XD     )
       DIMENSION XN(6),X(6),XD(6)
       DOUBLE PRECISION XN,C1(6),C2(6),C3(6)
       DO 1 I=1,6
     1 XN(I)=DBLE(X(I))
       TC=T
       CALL DERK
       DO 2 I=1,6
       C1(I)=H*XD(I)
       XN(I)=XN(I)+.5D0*C1(I)
     2 X(I)=SNGL(XN(I))
       T=TC+.5*H
       CALL DERK
       DO 3 I=1,6
       C2(I)=H*XD(I)
       XN(I)=XN(I)+.5D0*(C2(I)-C1(I))
     3 X(I)=SNGL(XN(I))
       CALL DERK
       DO 4 I=1,6
       C3(I)=H*XD(I)
       XN(I)=XN(I)+C3(I)-.5D0*C2(I)
     4 X(I)=SNGL(XN(I))
       T=TC+H
       CALL DERK
       DO 5 I=1,6
       XN(I)=XN(I)-C3(I)+.16666666666666667*(C1(I)+2.D0*(C2(I)+C3(I))
      1+H*XD(I))
     5 X(I)=SNGL(XN(I))
       RETURN
       END

$IBFTC DERK.  DECK
C
C      SUBROUTINE DERK PROVIDES THE DERIVATIVE LIST FOR THE INTERGRATION
C      ROUTINE IN KALMAN FILTER  -  TANGENT PLANE
C
       SUBROUTINE DERK
       COMMON C(999)
       REAL MU
       EQUIVALENCE        (C(141),X      ),(C(142),Y      ),(C(143),Z      ),
      1                   (C(144),VX     ),(C(145),VY     ),(C(146),VZ     ),
      2                   (C(147),ALPHA  ),(C(011),RE     ),
      3                   (C(021),WX     ),(C(022),WY     ),(C(023),WZ     ),
      4                   (C(012),MU     ),(C(013),WIE    ),(C(014),WIE2   ),
      5                   (C(151),XD     ),(C(152),YD     ),(C(153),ZD     ),
      6                   (C(154),VXD    ),(C(155),VYD    ),(C(156),VZD    )
       R=SQRT(X*X+Y*Y+Z*Z)
       V=SQRT(VX*VX+VY*VY+VZ*VZ)
       G=MU/(R**3)
       H=R-RE
       CALL ATMOS(H,RHO,GAMA)
       D=0.5*RHO   *V*ALPHA
       SUM=WX*X+WY*Y+WZ*Z
       XD=VX
       YD=VY
       ZD=VZ
       VXD=-G*X-D*VX-2.0*(WY*VZ-WZ*VY)-WX*SUM+X*WIE2
       VYD=-G*Y-D*VY-2.0*(WZ*VX-WX*VZ)-WY*SUM+Y*WIE2
       VZD=-G*Z-D*VZ-2.0*(WX*VY-WY*VX)-WZ*SUM+Z*WIE2
       RETURN
       END
```

```
$IBFTC PREDC.  DECK
C
C      SUBROUTINE PREDIC GENERATES PREDICTED VALUES OF POSITION
C      FROM THE PRESENT TIME - T - TO THE FINAL TIME -TF
C
       SUBROUTINE PREDII
       COMMON C(999)
       COMMON/PREDC/AA(500,4),AB(400,4),AC(300,4)
       INTEGER PKOUNT
       DIMENSION XK(7),XP(7),KOUNT(3)
       EQUIVALENCE      (C(001),TIME   ),(C(002),TF     ),(C(003),DT     ),
      1                 (C(982),KOUNT  ),(C(985),T      ),(C(986),HP     ),
      2                 (C(141),XK     ),(C(967),XP     ),(C(016),PKOUNT)
       IF(HP.LT.DT) HP=DT
       PKOUNT=1
       RETURN
       ENTRY PREDIC
       J=1
       T=TIME
       DO 1 I=1,7
     1 XP(I)=XK(I)
       CALL TRAJPI
       GO TO (3,5,7),PKOUNT
     2 J=J+1
       CALL TRAJP
       GO TO (3,5,7),PKOUNT
C
C      COMPUTE PREDICTION -A-
C
     3 AA(J,1)=T
       DO 4 K=2,4
     4 AA(J,K)=XP(K-1)
       GO TO 9
C
C      COMPUTE PREDICTION -B-
C
     5 AB(J,1)=T
       DO 6 K=2,4
     6 AB(J,K)=XP(K-1)
       GO TO 9
C
C      COMPUTE PREDICTION -C-
C
     7 AC(J,1)=T
       DO 8 K=2,4
     8 AC(J,K)=XP(K-1)
     9 IF(T.LT.TF) GO TO 2
       KOUNT(PKOUNT)=J
       PKOUNT=PKOUNT+1
       RETURN
       END
```

```
$IBFTC TRAJP.  DECK
C
C      INTEGRATION ROUTINE FOR THE FREDICTION SUBROUTINE
C      ADAMS-BASHFORTH  -  ADAMS-MOULTON  PREDICTOR-CORRECTOR WITH RUNGE-KUTTA
C
       SUBROUTINE TRAJPI
       COMMON C(999)
       DOUBLE PRECISION W
       DIMENSION D(6,5),W(6,5),Y(6),YD(6)
       EQUIVALENCE      (C(986),H      ),(C(985),X      ),(C(987),Y      ),
      1                 (C(994),YD    )
       DATA M/6/
       K=0
       K2=0
       DO 10 I=1,M
 10    W(I,1)=DBLE(Y(I))
       CALL DERP
       DO 1 I=1,6
     1 D(I,5)=YD(I)
       RETURN
       ENTRY TRAJP
 40    XC=X
       IF (K.NE.0) IF (K-2) 50,50,110
       XP=XC
       DO 45 I=1,M
 45    W(I,5)=W(I,1)
 50    K1=4-K
       DO 70 I=1,M
       DO 60 J=K1,4
 60    D(I,J)=D(I,J+1)
       W(I,2)=H*D(I,4)
       W(I,1)=W(I,1)+.5D0*W(I,2)
 70    Y(I)=SNGL(W(I,1))
       X=XC+.5*H
       CALL DERP
       DO 2 I=1,6
     2 D(I,5)=YD(I)
       DO 80 I=1,M
       W(I,3)=H*D(I,5)
       W(I,1)=W(I,1)+.5D0*(W(I,3)-W(I,2))
 80    Y(I)=SNGL(W(I,1))
       CALL DERP
       DO 3 I=1,6
     3 D(I,5)=YD(I)
       DO 90 I=1,M
       W(I,4)=H*D(I,5)
       W(I,1)=W(I,1)+W(I,4)-.5D0*W(I,3)
 90    Y(I)=SNGL(W(I,1))
       X=XC+H
       CALL DERP
       DO 4 I=1,6
     4 D(I,5)=YD(I)
       DO 100 I=1,M
       W(I,1)=W(I,1)-W(I,4)+.16666666666666667*(W(I,2)+2.D0*(W(I,3)+W(I,4
      1))+H*D(I,5))
 100   Y(I)=SNGL(W(I,1))
       K=K+1
       K1=K
       CALL DERP
       DO 5 I=1,6
     5 D(I,5)=YD(I)
       RETURN
 110   DO 130 I=1,M
       W(I,2)=W(I,1)
       DO 120 J=1,4
 120   D(I,J)=D(I,J+1)
       W(I,3)=W(I,2)+.41666666666666667D-1*H*(55.*D(I,4)-59.*D(I,3)+37.*D
      1(I,2)-9.*D(I,1))
 130   Y(I)=SNGL(W(I,3))
       X=XC+H
       CALL DERP
       DO 6 I=1,6
     6 D(I,5)=YD(I)
```

```
      DO 140 I=I,M
      W(I,1)=W(I,2)+.41666666666666667D-1*H*(9.*D(I,5)+19.*D(I,4)-5.*D(I
     1,3)+D(I,2))
  140 Y(I)=SNGL(W(I,1))
      CALL DERT
      DO 7 I=1,6
    7 D(I,5)=YD(I)
      RETURN
      END
```

```
$IBFTC DERT.   DECK
C
C     SUBROUTINE DERT PROVIDES THE DERIVATIVE LIST FOR THE INTEGRATION
C     ROUTINE FOR THE REFERENCE TRAJECTORY  -  EARTH COORDINATES
C
      SUBROUTINE DERT
      COMMON C(999)
      REAL MU
      EQUIVALENCE      (C(101),X    ),(C(102),Y     ),(C(103),Z     ),
     1                 (C(104),VX   ),(C(105),VY    ),(C(106),VZ    ),
     2                 (C(107),BETA ),(C(011),RE    ),
     3                 (C(021),WX   ),(C(022),WY    ),(C(023),WZ    ),
     4                 (C(012),MU   ),(C(013),WIE   ),(C(014),WIE2  ),
     5                 (C(111),XD   ),(C(112),YD    ),(C(113),ZD    ),
     6                 (C(114),VXD  ),(C(115),VYD   ),(C(116),VZD   )
      R=SQRT(X*X+Y*Y+Z*Z)
      V=SQRT(VX*VX+VY*VY+VZ*VZ)
      G=MU/(R**3)
      H=R-RE
      CALL ATMOS(H,RHO,GAMA)
      D=0.5*RHO*V/BETA
      XD=VX
      YD=VY
      ZD=VZ
      VXD=-G*X-D*VX+2.0*WIE*VY+X*WIE2
      VYD=-G*Y-D*VY-2.0*WIE*VX+Y*WIE2
      VZD=-G*Z-D*VZ
      RETURN
      END
```

```
$IBFTC COMPR.  DECK
C
C     SUBROUTINE COMPAR COMPUTES THE DIFFERENCE BETWEEN THE ACTUAL VALUES
C     OF POSITION AND VELOCITY AND THE ESTIMATED AND PREDICTED VALUES
C
      SUBROUTINE COMPAI
      COMMON C(999)
      COMMON/PREDC/AA(500,4),AB(400,4),AC(300,4)
      COMMON/CALCOM/TT(700),DR(700),PDR(700),DV(700),PDV(700),EB(700),I,
     1TA(500),PDRA(500),J,TB(400),PDRB(400),TC(300),PDRC(300),K
      INTEGER PKOUNT
      EQUIVALENCE     (C(120),XTM   ),(C(121),YTM   ),(C(122),ZTM   ),
     1(C(107),BETA  ),(C(123),VXTM  ),(C(124),VYTM  ),(C(125),VZTM  ),
     2                (C(141),EXTM  ),(C(142),EYTM  ),(C(143),EZTM  ),
     3(C(140),EBETA ),(C(144),EVXTM ),(C(145),EVYTM ),(C(146),EVZTM ),
     4                (C(024),DELX  ),(C(025),DELY  ),(C(026),DELZ  ),
     5(C(030),DBETA ),(C(027),DELVX ),(C(028),DELVY ),(C(029),DELVZ ),
     6                (C(118),DELR  ),(C(119),DELV  ),(C(016),PKOUNT),
     7                (C(976),DELPRA),(C(977),DELPRB),(C(978),DELPRC),
     8                (C(138),SEPR  ),(C(139),SEPV  ),(C(001),T     )
      DTH=0.005
      I=0
      J=0
      K=0
      L=0
      RETURN
      ENTRY COMPAR
C
C     COMPUTE ERRORS IN ESTIMATION
C
      DELX=EXTM-XTM
      DELY=EYTM-YTM
      DELZ=EZTM-ZTM
      DELVX=EVXTM-VXTM
      DELVY=EVYTM-VYTM
      DELVZ=EVZTM-VZTM
      DELR=SQRT(DELX*DELX+DELY*DELY+DELZ*DELZ)
      DELV=SQRT(DELVX*DELVX+DELVY*DELVY+DELVZ*DELVZ)
      DBETA=EBETA-BETA
C
C     LOAD ARRAYS FOR PLOTTING
C
      I=I+1
      TT(I)=T
      DR(I)=DELR
      DV(I)=DELV
      EB(I)=EBETA
      GO TO (7,5,3,1),PKOUNT
C
C     COMPUTE ERRORS IN PREDICTION  -C-
C
    1 L=L+1
      DIFF=T-AC(L,1)
      IF(ABS(DIFF).GT.DTH) GO TO 2
      DELPRC=SQRT((AC(L,2)-XTM)**2+(AC(L,3)-YTM)**2+(AC(L,4)-ZTM)**2)
C     LOAD ARRAYS FOR PLOTTING
      TC(L)=T
      PDRC(L)=DELPRC
    2 IF(DIFF.GT.0.0) GO TO 1
C
C     COMPUTE ERRORS IN PREDICTION  -B-
C
    3 K=K+1
      DIFF=T-AB(K,1)
      IF(ABS(DIFF).GT.DTH) GO TO 4
      DELPRB=SQRT((AB(K,2)-XTM)**2+(AB(K,3)-YTM)**2+(AB(K,4)-ZTM)**2)
C     LOAD ARRAYS FOR PLOTTING
      TB(K)=T
      PDRB(K)=DELPRB
    4 IF(DIFF.GT.0.0) GO TO 3
C
C     COMPUTE ERRORS IN PREDICTION ,-A-
C
```

```
    5 J=J+1
      DIFF=T-AA(J,1)
      IF(ABS(DIFF).GT.DTH) GO TO 6
      DELPRA=SQRT((AA(J,2)-XTM)**2+(AA(J,3)-YTM)**2+(AA(J,4)-ZTM)**2)
C     LOAD ARRAYS FOR PLOTTING
      TA(J)=T
      PDRA(J)=DELPRA
    6 IF(DIFF.GT.0.0) GO TO 5
    7 RETURN
      END
```

```
$IBFTC NOISE.  DECK
C
C     SUBROUTINE NOISE GENERATES GAUSSIAN NOISE
C
      SUBROUTINE NOISEI
      COMMON C(999)
      INTEGER RNDMNO(5),IX(5)
      EQUIVALENCE     (C(490),NORNDM),(C(491),RNDMNO),(C(003),DT     )
      IF(NORNDM.EQ.0) RETURN
      DO 1 I=1,NORNDM
      J=RNDMNO(I)
      IF(C(J+2).LE.0.0) C(J+2)=0.0000001
      C(J+3)=2.7182818**(-DT/C(J+2))
      C(J+4)=C(J+1)*SQRT(1.0-C(J+3)*C(J+3))
      IXI=C(J)
      CALL RANDU(IXI,IY,V)
      IX(I)=IY
    1 C(J+6)=C(J+1)*V
      RETURN
      ENTRY NOISE
      IF(NORNDM.EQ.0) RETURN
      DO 2 I=1,NORNDM
      J=RNDMNO(I)
      IXI=IX(I)
      SUM=0.0
      DO 3 K=1,12
      CALL RANDU(IXI,IY,V)
      IXI=IY
    3 SUM=SUM+V
      X=SUM-6.0
      IX(I)=IXI
      C(J+5)=C(J+6)
    2 C(J+6)=C(J+4)*X+C(J+3)*C(J+5)
      RETURN
      END
```

```
$IBFTC RANDU.  DECK
C
C     ...............................................................
C
C        SUBROUTINE RANDU
C
C        PURPOSE
C           COMPUTES UNIFORMLY DISTRIBUTED RANDOM REAL NUMBERS BETWEEN
C           0 AND 1.0 AND RANDOM INTEGERS BETWEEN ZERO AND
C           2**31. EACH ENTRY USES AS INPUT AN INTEGER RANDOM NUMBER
C           AND PRODUCES A NEW INTEGER AND REAL RANDOM NUMBER.
C
C        USAGE
C           CALL RANDU(IX,IY,YFL)
C
C        DESCRIPTION OF PARAMETERS
C           IX - FOR THE FIRST ENTRY THIS MUST CONTAIN ANY ODD INTEGER
C                NUMBER WITH NINE OR LESS DIGITS. AFTER THE FIRST ENTRY,
C                IX SHOULD BE THE PREVIOUS VALUE OF IY COMPUTED BY THIS
C                SUBROUTINE.
C           IY - A RESULTANT INTEGER RANDOM NUMBER REQUIRED FOR THE NEXT
C                ENTRY TO THIS SUBROUTINE. THE RANGE OF THIS NUMBER IS
C                BETWEEN ZERO AND 2**31
C           YFL- THE RESULTANT UNIFORMLY DISTRIBUTED, FLOATING POINT,
C                RANDOM NUMBER IN THE RANGE 0 TO 1.0
C
C        REMARKS
C           THIS SUBROUTINE IS SPECIFIC TO SYSTEM/360
C           THIS SUBROUTINE WILL PRODUCE 2**29 TERMS
C           BEFORE REPEATING
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        METHOD
C           POWER RESIDUE METHOD DISCUSSED IN IBM MANUAL C20-8011,
C           RANDOM NUMBER GENERATION AND TESTING
C
C     ...............................................................
C
      SUBROUTINE RANDU(IX,IY,YFL)
      IY=IX*262147
      IF(IY.LT.0) IY=(IY+34359738367)+1
      YFL=IY
      YFL=YFL*.29103383046E-10
      RETURN
      END
```

```
$IBFTC ATMOS.  DECK
C
C      SUBROUTINE ATMOS PROVIDES AIR DENSITY AND RATE-OF-CHANGE OF AIR DENSITY AS
C      A FUNCTION  OF ALTITUDE.  AIR DENSITY IS ACCURATE TO WITHIN 2.0 PER-CENT
C      OVER AN ALTITUDE RANGE OF -10,000 FEET TO +2,000,000 FEET AND TO WITHIN
C      0.2 PER-CENT IN THE RANGE -1,000 FEET TO 40,000 FEET.  INTERPOLATION IS
C      LINEAR.  TABLE ENTRIES ARE FROM THE 1959 ARDC MODEL ATMOSPHERE.
C
C          -10,000  FEET     1.0150  E-01  LBS/CU.FT.
C           -5,000  FEET     8.8310  E-02  LBS/CU.FT.
C           -1,000  FEET     7.8738  E-02  LBS/CU.FT.
C          SEA LEVEL         7.6475  E-02  LBS/CU.FT.
C            1,000  FEET     7.4262  E-02  LBS/CU.FT.
C            2,000  FEET     7.2099  E-02  LBS/CU.FT.
C            4,000  FEET     6.7918  E-02  LBS/CU.FT.
C            6,000  FEET     6.3926  E-02  LBS/CU.FT.
C            8,000  FEET     6.0116  E-02  LBS/CU.FT.
C           10,000  FEET     5.6483  E-02  LBS/CU.FT.
C           12,000  FEET     5.3022  E-02  LBS/CU.FT.
C           14,000  FEET     4.9725  E-02  LBS/CU.FT.
C           16,000  FEET     4.6589  E-02  LBS/CU.FT.
C           18,000  FEET     4.3606  E-02  LBS/CU.FT.
C           20,000  FEET     4.0773  E-02  LBS/CU.FT.
C           22,000  FEET     3.8083  E-02  LBS/CU.FT.
C           24,000  FEET     3.5531  E-02  LBS/CU.FT.
C           26,000  FEET     3.3113  E-02  LBS/CU.FT.
C           28,000  FEET     3.0823  E-02  LBS/CU.FT.
C           30,000  FEET     2.8657  E-02  LBS/CU.FT.
C           32,000  FEET     2.6609  E-02  LBS/CU.FT.
C           34,000  FEET     2.4676  E-02  LBS/CU.FT.
C           36,000  FEET     2.2852  E-02  LBS/CU.FT.
C           38,000  FEET     2.0794  E-02  LBS/CU.FT.
C           40,000  FEET     1.8895  E-02  LBS/CU.FT.
C           45,000  FEET     1.4873  E-02  LBS/CU.FT.
C           50,000  FEET     1.1709  E-02  LBS/CU.FT.
C           55,000  FEET     9.2185  E-03  LBS/CU.FT.
C           60,000  FEET     7.2588  E-03  LBS/CU.FT.
C           65,000  FEET     5.7164  E-03  LBS/CU.FT.
C           70,000  FEET     4.5022  E-03  LBS/CU.FT.
C           75,000  FEET     3.5463  E-03  LBS/CU.FT.
C           80,000  FEET     2.7937  E-03  LBS/CU.FT.
C           85,000  FEET     2.1784  E-03  LBS/CU.FT.
C           90,000  FEET     1.6901  E-03  LBS/CU.FT.
C           95,000  FEET     1.3182  E-03  LBS/CU.FT.
C          100,000  FEET     1.0332  E-03  LBS/CU.FT.
C          110,000  FEET     6.4392  E-04  LBS/CU.FT.
C          120,000  FEET     4.0851  E-04  LBS/CU.FT.
C          130,000  FEET     2.6349  E-04  LBS/CU.FT.
C          140,000  FEET     1.7258  E-04  LBS/CU.FT.
C          150,000  FEET     1.1468  E-04  LBS/CU.FT.
C          160,000  FEET     7.8276  E-05  LBS/CU.FT.
C          170,000  FEET     5.4467  E-05  LBS/CU.FT.
C          180,000  FEET     3.8700  E-05  LBS/CU.FT.
C          190,000  FEET     2.7836  E-05  LBS/CU.FT.
C          200,000  FEET     1.9634  E-05  LBS/CU.FT.
C          210,000  FEET     1.3659  E-05  LBS/CU.FT.
C          220,000  FEET     9.2807  E-06  LBS/CU.FT.
C          230,000  FEET     6.1583  E-06  LBS/CU.FT.
C          240,000  FEET     3.9784  E-06  LBS/CU.FT.
C          250,000  FEET     2.493   E-06  LBS/CU.FT.
C          260,000  FEET     1.508   E-06  LBS/CU.FT.
C          270,000  FEET     8.343   E-07  LBS/CU.FT.
C          280,000  FEET     4.522   E-07  LBS/CU.FT.
C          290,000  FEET     2.453   E-07  LBS/CU.FT.
C          300,000  FEET     1.327   E-07  LBS/CU.FT.
C          310,000  FEET     6.880   E-08  LBS/CU.FT.
C          320,000  FEET     3.724   E-08  LBS/CU.FT.
C          330,000  FEET     2.093   E-08  LBS/CU.FT.
C          340,000  FEET     1.216   E-08  LBS/CU.FT.
C          350,000  FEET     7.282   E-09  LBS/CU.FT.
C        2,000,000  FEET     0.000         LBS/CU.FT.
C      THE AIR DENSITY ABOVE 2,000,000 FEET IS ASSUMED TO BE ZERO.
C
```

```
      SUBROUTINE ATMOSI
      DIMENSION  PTAB(63), ATAB(63),GTAB(62)
      DATA ATAB/-1.0E4,-5.0E3,-1.0E3,0.0E0,1.0E3,2.0E3,4.0E3,6.0E3,
     18.0E3,1.0E4,1.2E4,1.4E4,1.6E4,1.8E4,2.0E4,2.2E4,2.4E4,2.6E4,
     22.8E4,3.0E4,3.2E4,3.4E4,3.6E4,3.8E4,4.0E4,4.5E4,5.0E4,5.5E4,
     36.0E4,6.5E4,7.0E4,7.5E4,8.0E4,8.5E4,9.0E4,9.5E4,1.0E5,1.1E5,
     41.2E5,1.3E5,1.4E5,1.5E5,1.6E5,1.7E5,1.8E5,1.9E5,2.0E5,2.1E5,
     52.2E5,2.3E5,2.4E5,2.5E5,2.6E5,2.7E5,2.8E5,2.9E5,3.0E5,3.1E5,
     63.2E5,3.3E5,3.4E5,3.5E5,2.0E6/
      DATA PTAB/1.0150E-01,8.8310E-02,7.8738E-02,7.6475E-02,7.4262E-02,
     17.2099E-02,6.7918E-02,6.3926E-02,6.0116E-02,5.6483E-02,5.3022E-02,
     24.9725E-02,4.6589E-02,4.3606E-02,4.0773E-02,3.8083E-02,3.5531E-02,
     33.3113E-02,3.0823E-02,2.8657E-02,2.6609E-02,2.4676E-02,2.2852E-02,
     42.0794E-02,1.8895E-02,1.4873E-02,1.1709E-02,9.2185E-03,7.2588E-03,
     55.7164E-03,4.5022E-03,3.5463E-03,2.7937E-03,2.1784E-03,1.6901E-03,
     61.3182E-03,1.0332E-03,6.4392E-04,4.0851E-04,2.6349E-04,1.7258E-04,
     71.1468E-04,7.8276E-05,5.4467E-05,3.8700E-05,2.7836E-05,1.9684E-05,
     81.3659E-05,9.2807E-06,6.1583E-06,3.9784E-06,2.4930E-06,1.5080E-06,
     98.3430E-07,4.5220E-07,2.4530E-07,1.3270E-07,6.8800E-08,3.7240E-08,
     12.0930E-08,1.2160E-08,7.2820E-09,0.0E0/,M/1/
      DO 10 I=1,62
   10 GTAB(I)=(PTAB(I+1)-PTAB(I))/(ATAB(I+1)-ATAB(I))
      RETURN
      ENTRY ATMOS(H,RHO,PRHO)
      IF (H .GE. ATAB(63)) GO TO 3
    1 IF (H - ATAB(M+1)) 7,2,4
    2 RHO = PTAB(M+1)
      GO TO 9
    3 RHO = 0.
      PRHO=0.0
      GO TO 9
    4 IF (H - ATAB(M+2)) 8,6,5
    5 M = M+1
      GO TO 4
    6 M = M + 1
      GO TO 2
    7 M = M - 1
      GO TO 1
    8 RHO = PTAB(M+1) + (H - ATAB(M+1))/(ATAB(M+2) - ATAB(M+1))*(PTAB
     1(M+2) - PTAB(M+1))
      PRHO=GTAB(M+1)
    9 RETURN
      END
```

```
$IBFTC INPUT.  DECK
C
C       SUBROUTINE INPUT - READS ALL INPUT DATA
C
      SUBROUTINE INPUT
      COMMON C(999)
      INTEGER OUTNO,RNDMNO(5)
      DIMENSION ONAME1(50),ONAME2(50),OUTNO(50),LISTNO(50),VALUE(50),
      EQUIVALENCE      (C(490),NORNDM),(C(499),NOLIST),(C(500),NOOUT ),
     1                 (C(501),ONAME1),(C(551),ONAME2),(C(601),OUTNO ),
     2                 (C(651),LISTNO),(C(701),VALUE ),(C(491),RNDMNO)
      WRITE(6,600)
  600 FORMAT(1H1,4X,10HINPUT DATA//)
  100 READ (5,500) IR1,ALPHA1,ALPHA2,ALPHA3,IR2,VR1,VR2
  500 FORMAT(I2,3A6,I5,5X,2E15.0)
      WRITE(6,601) IR1,ALPHA1,ALPHA2,ALPHA3,IR2,VR1,VR2
  601 FORMAT(5X,I2,3A6,I5,5X,1P2E15.7)
      GO TO (1,2,3,4,5,6),IR1
    1 GO TO 100
    2 GO TO 100
    3 C(IR2)=VR1
      IF(VR2.EQ.0.0)  GO TO 100
      NOLIST=NOLIST+1
      LISTNO(NOLIST)=IR2
      VALUE(NOLIST)=VR1
      GO TO 100
    4 NOOUT=NOOUT+1
      ONAME1(NOOUT)=ALPHA2
      ONAME2(NOOUT)=ALPHA3
      OUTNO(NOOUT)=IR2
      GO TO 100
    5 GO TO 100
    6 IF(IR2.EQ.0) RETURN
      DO 7 I=1,IR2
      READ(5,501) J,X,NAME1,NAME2,SIGMA,NAME3,NAME4,TAU
  501 FORMAT(I5,E15.0,2A5,E15.0,2A5,E15.0)
      WRITE(6,602) J,X,NAME1,NAME2,SIGMA,NAME3,NAME4,TAU
  602 FORMAT(5X,I5,F15.3,2A5,1PE15.7,2A5,1PE15.7)
      NORNDM=NORNDM+1
      RNDMNO(I)=J
      C(J)=X
      C(J+1)=SIGMA
    7 C(J+2)=TAU
      RETURN
      END
```

```
$IBFTC OUPTI.  DECK
C
C      SUBROUTINE OUTPUT - OUTPUTS DATA
C
       SUBROUTINE OUPTI
       COMMON C(999)
       INTEGER DTCNT,PGCNT,OUTNO
       DIMENSION ONAME1(50),ONAME2(50),OUTNO(50),B(50)
       EQUIVALENCE     (C(001),T      ),(C(004),CPP    ),(C(486),PCNT  ),
      1(C(487),DTCNT ),(C(488),PGCNT ),(C(489),ITCNT ),(C(005),DOC   ),
      2(C(500),NOOUT ),(C(501),ONAME1),(C(551),ONAME2),(C(601),OUTNO )
       ITCNT = DOC + 1.0
       PCNT = T-0.000001
       PGCNT = 1
       DTCNT = (NOOUT + 4)/5
       GO TO 100
       ENTRY OUTPUT
   100 IF(ITCNT.GT.6) GO TO 1
       ITCNT=ITCNT+1
       WRITE (6,600) (I,C(I),C(I+1),C(I+2),C(I+3),C(I+4),C(I+5),C(I+6),
      1 C(I+7),I=1,472,8)
   600 FORMAT(1H1,5X,14HCOMMON LISTING/(I5,2X,1P8E15.7))
       PGCNT=1
     1 IF(T.LT.PCNT) RETURN
       PCNT=PCNT+CPP
       IF(PGCNT.NE.1) GO TO 3
     2 WRITE(6,601)  (ONAME1(I),ONAME2(I),I=1,NOOUT)
   601 FORMAT (1H1,5X,4HTIME,5X,5(8X,2A6)/ (23X,2A6,8X,2A6,8X,2A6,8X,
      12A6,8X,2A6) )
       PGCNT=2*DTCNT+4
     3 IF(PGCNT.GE.62) GO TO 2
       DO 4 I=1,NOOUT
       J=OUTNO(I)
     4 B(I)=C(J)
       WRITE(6,603) T,(B(I),I=1,NOOUT)
   603 FORMAT(////2X,F15.7,1P5E20.7/(17X,1P5E20.7))
       PGCNT = PGCNT + DTCNT + 4
       RETURN
       END
```

```
$IBFTC RESET.  DECK
C
C      SUBROUTINE RESET RESETS SELECTED INPUT DATA FOR REPEATED RUNS.
C
       SUBROUTINE RESET
       COMMON C(999)
       EQUIVALENCE     (C(499),NOLIST),(C(651),LISTNO),(C(701),VALUE )
       DIMENSION LISTNO(50),VALUE(50)
       IF (NOLIST .EQ. 0) RETURN
       DO 1 I = 1, NOLIST
       J = LISTNO(I)
     1 C(J) = VALUE(I)
       RETURN
       END
```

```
$IBFTC MFSDO   DECK
C
C     ..................................................................
C
C        SUBROUTINE MFSD
C
C        PURPOSE
C           FACTOR A GIVEN SYMMETRIC POSITIVE DEFINITE MATRIX
C
C        USAGE
C           CALL MFSD(A,N,EPS,IER)
C
C        DESCRIPTION OF PARAMETERS
C           A      - UPPER TRIANGULAR PART OF THE GIVEN SYMMETRIC
C                    POSITIVE DEFINITE N BY N COEFFICIENT MATRIX.
C                    ON RETURN A CONTAINS THE RESULTANT UPPER
C                    TRIANGULAR MATRIX.
C           N      - THE NUMBER OF ROWS (COLUMNS) IN GIVEN MATRIX.
C           EPS    - AN INPUT CONSTANT WHICH IS USED AS RELATIVE
C                    TOLERANCE FOR TEST ON LOSS OF SIGNIFICANCE.
C           IER    - RESULTING ERROR PARAMETER CODED AS FOLLOWS
C                    IER=0  - NO ERROR
C                    IER=-1 - NO RESULT BECAUSE OF WRONG INPUT PARAME-
C                             TER N OR BECAUSE SOME RADICAND IS NON-
C                             POSITIVE (MATRIX A IS NOT POSITIVE
C                             DEFINITE, POSSIBLY DUE TO LOSS OF SIGNI-
C                             FICANCE)
C                    IER=K  - WARNING WHICH INDICATES LOSS OF SIGNIFI-
C                             CANCE. THE RADICAND FORMED AT FACTORIZA-
C                             TION STEP K+1 WAS STILL POSITIVE BUT NO
C                             LONGER GREATER THAN ABS(EPS*A(K+1,K+1)).
C
C        REMARKS
C           THE UPPER TRIANGULAR PART OF GIVEN MATRIX IS ASSUMED TO BE
C           STORED COLUMNWISE IN N*(N+1)/2 SUCCESSIVE STORAGE LOCATIONS.
C           IN THE SAME STORAGE LOCATIONS THE RESULTING UPPER TRIANGU-
C           LAR MATRIX IS STORED COLUMNWISE TOO.
C           THE PROCEDURE GIVES RESULTS IF N IS GREATER THAN 0 AND ALL
C           CALCULATED RADICANDS ARE POSITIVE.
C           THE PRODUCT OF RETURNED DIAGONAL TERMS IS EQUAL TO THE
C           SQUARE-ROOT OF THE DETERMINANT OF THE GIVEN MATRIX.
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        METHOD
C           SOLUTION IS DONE USING THE SQUARE-ROOT METHOD OF CHOLESKY.
C           THE GIVEN MATRIX IS REPRESENTED AS PRODUCT OF TWO TRIANGULAR
C           MATRICES, WHERE THE LEFT HAND FACTOR IS THE TRANSPOSE OF
C           THE RETURNED RIGHT HAND FACTOR.
C
C     ..................................................................
C
      SUBROUTINE MFSD(A,N,EPS,IER)
C
C
      DIMENSION A(1)
      DOUBLE PRECISION DPIV,DSUM
C
C        TEST ON WRONG INPUT PARAMETER N
      IF(N-1) 12,1,1
    1 IER=0
C
C        INITIALIZE DIAGONAL-LOOP
      KPIV=0
      DO 11 K=1,N
      KPIV=KPIV+K
      IND=KPIV
      LEND=K-1
C
C        CALCULATE TOLERANCE
      TOL=ABS(EPS*A(KPIV))
C
```

```
C        START FACTORIZATION-LOOP OVER K-TH ROW
      DO 11 I=K,N
      DSUM=0.D0
      IF(LEND) 2,4,2
C
C        START INNER LOOP
    2 DO 3 L=1,LEND
      LANF=KPIV-L
      LIND=IND-L
    3 DSUM=DSUM+DBLE(A(LANF)*A(LIND))
C        END OF INNER LOOP
C
C        TRANSFORM ELEMENT A(IND)
    4 DSUM=DBLE(A(IND))-DSUM
      IF(I-K) 10,5,10
C
C        TEST FOR NEGATIVE PIVOT ELEMENT AND FOR LOSS OF SIGNIFICANCE
    5 IF(SNGL(DSUM)-TOL) 6,6,9
    6 IF(DSUM) 12,12,7
    7 IF(IER) 8,8,9
    8 IER=K-1
C
C        COMPUTE PIVOT ELEMENT
    9 DPIV=DSQRT(DSUM)
      A(KPIV)=DPIV
      DPIV=1.D0/DPIV
      GO TO 11
C
C        CALCULATE TERMS IN ROW
   10 A(IND)=DSUM*DPIV
   11 IND=IND+I
C
C        END OF DIAGONAL-LOOP
      RETURN
   12 IER=-1
      RETURN
      END
```

```
$IBFTC SINVO   DECK
C
C     ...............................................................
C
C        SUBROUTINE SINV
C
C        PURPOSE
C           INVERT A GIVEN SYMMETRIC POSITIVE DEFINITE MATRIX
C
C        USAGE
C           CALL SINV(A,N,EPS,IER)
C
C        DESCRIPTION OF PARAMETERS
C           A      - UPPER TRIANGULAR PART OF THE GIVEN SYMMETRIC
C                    POSITIVE DEFINITE N BY N COEFFICIENT MATRIX.
C                    ON RETURN A CONTAINS THE RESULTANT UPPER
C                    TRIANGULAR MATRIX.
C           N      - THE NUMBER OF ROWS (COLUMNS) IN GIVEN MATRIX.
C           EPS    - AN INPUT CONSTANT WHICH IS USED AS RELATIVE
C                    TOLERANCE FOR TEST ON LOSS OF SIGNIFICANCE.
C           IER    - RESULTING ERROR PARAMETER CODED AS FOLLOWS
C                    IER=0  - NO ERROR
C                    IER=-1 - NO RESULT BECAUSE OF WRONG INPUT PARAME-
C                             TER N OR BECAUSE SOME RADICAND IS NON-
C                             POSITIVE (MATRIX A IS NOT POSITIVE
C                             DEFINITE, POSSIBLY DUE TO LOSS OF SIGNI-
C                             FICANCE)
C                    IER=K  - WARNING WHICH INDICATES LOSS OF SIGNIFI-
C                             CANCE. THE RADICAND FORMED AT FACTORIZA-
C                             TION STEP K+1 WAS STILL POSITIVE BUT NO
C                             LONGER GREATER THAN ABS(EPS*A(K+1,K+1)).
C
C        REMARKS
C           THE UPPER TRIANGULAR PART OF GIVEN MATRIX IS ASSUMED TO BE
C           STORED COLUMNWISE IN N*(N+1)/2 SUCCESSIVE STORAGE LOCATIONS.
C           IN THE SAME STORAGE LOCATIONS THE RESULTING UPPER TRIANGU-
C           LAR MATRIX IS STORED COLUMNWISE TOO.
C           THE PROCEDURE GIVES RESULTS IF N IS GREATER THAN 0 AND ALL
C           CALCULATED RADICANDS ARE POSITIVE.
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           MFSD
C
C        METHOD
C           SOLUTION IS DONE USING THE FACTORIZATION BY SUBROUTINE MFSD.
C
C     ...............................................................
C
      SUBROUTINE SINV(A,N,EPS,IER)
C
C
      DIMENSION A(1)
      DOUBLE PRECISION DIN,WORK
C
C        FACTORIZE GIVEN MATRIX BY MEANS OF SUBROUTINE MFSD
C        A = TRANSPOSE(T) * T
      CALL MFSD(A,N,EPS,IER)
      IF(IER) 9,1,1
C
C        INVERT UPPER TRIANGULAR MATRIX T
C        PREPARE INVERSION-LOOP
    1 IPIV=N*(N+1)/2
      IND=IPIV
C
C        INITIALIZE INVERSION-LOOP
      DO 6 I=1,N
      DIN=1.D0/DBLE(A(IPIV))
      A(IPIV)=DIN
      MIN=N
      KEND=I-1
      LANF=N-KEND
      IF(KEND) 5,5,2
    2 J=IND
```

```
C
C        INITIALIZE ROW-LOOP
      DO 4 K=1,KEND
      WORK=0.D0
      MIN=MIN-1
      LHOR=IPIV
      LVER=J
C
C        START INNER LOOP
      DO 3 L=LANF,MIN
      LVER=LVER+1
      LHOR=LHOR+L
    3 WORK=WORK+DBLE(A(LVER)*A(LHOR))
C        END OF INNER LOOP
C
      A(J)=-WORK*DIN
    4 J=J-MIN
C        END OF ROW-LOOP
C
    5 IPIV=IPIV-MIN
    6 IND=IND-1
C        END OF INVERSION-LOOP
C
C        CALCULATE INVERSE(A) BY MEANS OF INVERSE(T)
C        INVERSE(A) = INVERSE(T) * TRANSPOSE(INVERSE(T))
C        INITIALIZE MULTIPLICATION-LOOP
      DO 8 I=1,N
      IPIV=IPIV+I
      J=IPIV
C
C        INITIALIZE ROW-LOOP
      DO 8 K=I,N
      WORK=0.D0
      LHOR=J
C
C        START INNER LOOP
      DO 7 L=K,N
      LVER=LHOR+K-I
      WORK=WORK+DBLE(A(LHOR)*A(LVER))
    7 LHOR=LHOR+L
C        END OF INNER LOOP
C
      A(J)=WORK
    8 J=J+K
C        END OF ROW- AND MULTIPLICATION-LOOP
C
    9 RETURN
      END
```

```
C******************************** COMMON LISTING ********************************
C
C
C(  1)       T
C(  2)       TF
C(  3)       DT
C(  4)       CCP
C(  5)       DOC
C(  6)       STEP
C(  7)
C(  8)       TTSKF
C(  9)       TK
C( 10)       DT2
C( 11)       RE
C( 12)       MU
C( 13)       WIE
C( 14)       WIE2
C( 15)       EPS
C( 16)       PKOUNT
C( 17)       PTIME(1)
C( 18)       PTIME(2)
C( 19)       PTIME(3)
C( 20)       PTIME(4
C( 21)       WX
C( 22)       WY
C( 23)       WZ
C( 24)       DELX
C( 25)       DELY
C( 26)       DELZ
C( 27)       DELVX
C( 28)       DELVY
C( 29)       DELVZ
C( 30)       DBETA
C( 31) THRU C( 39)      CET11 THRU CET33  STORED COLUMN WISE
C( 40)       AZD
C( 41) THRU C(C49)      CAE11 THRU CAE33  STORED COLUMN WISE
C( 50)       ELD
C( 51) THRU C( 59)      CAT11 THRU CAT33  STORED COLUMN WISE
C( 60)
C( 61)
C( 62)
C( 63)
C( 64)
C( 65)
C( 65)
C( 67)
C( 68)
C( 69)
C( 70)       AZ
C( 71)
C( 72)
C( 73)
C( 74)
C( 75)
C( 76)
C( 77)
C( 78)
C( 79)
C( 80)       EL
C( 81)
C( 82)
C( 83)
C( 84)
C( 85)
C( 86)
C( 87)
C( 88)
C( 89)
C( 90)       RA
C( 91)
C( 92)
C( 93)
C( 94)
```

C(95)	
C(96)	
C(97)	
C(98)	
C(99)	
C(100)	RR
C(101)	XEM
C(102)	YEM
C(103)	ZEM
C(104)	VXEM
C(105)	VYEM
C(106)	VZEM
C(107)	BETA
C(108)	HM
C(109)	VM
C(110)	Q
C(111)	XDEM
C(112)	YDEM
C(113)	ZDEM
C(114)	VXDEM
C(115)	VYDEM
C(116)	VZDEM
C(117)	
C(118)	DELR
C(119)	DELV
C(120)	XTM
C(121)	YTM
C(122)	ZTM
C(123)	VXTM
C(124)	VYTM
C(125)	VZTM
C(126)	LAT
C(127)	LONG
C(128)	HP
C(129)	HEAD
C(130)	VP
C(131)	XEP
C(132)	YEP
C(133)	ZEP
C(134)	VXEP
C(135)	VYEP
C(136)	VZEP
C(137)	GAMMA
C(138)	SEPR
C(139)	SEPV
C(140)	EBETA
C(141)	EXTM
C(142)	EYTM
C(143)	EZTM
C(144)	EVXTM
C(145)	EVYTM
C(146)	EVZTM
C(147)	ALPHA
C(148)	EHM
C(149)	EVM
C(150)	
C(151)	EXDTM
C(152)	EYDTM
C(153)	EZDTM
C(154)	EVXDTM
C(155)	EVYDTM
C(156)	EVZDTM
C(157)	Z(1)
C(158)	Z(2)
C(159)	Z(3)
C(160)	Z(4)
C(161)	DXEST(1)
C(162)	DXEST(2)
C(163)	DXEST(3)
C(164)	DXEST(4)
C(165)	DXEST(5)
C(166)	DXEST(6)
C(167)	DXEST(7)

```
C(168)          SIGAZ
C(169)          SIGEL
C(170)          SIGRA
C(171)          SIGRR
C(172)          D
C(173) THRU C(200)      K(1,1) THRU K(7,4)  STORED COLUMN WISE
C(201) THRU C(249)      F(1,1) THRU F(7,7)  STORED COLUMN WISE
C(250)
C(251) THRU C(299)      PHI(1,1) THRU PHI(7,7)  STORED COLUMN WISE
C(300)
C(301) THRU C(349)      PP(1,1) THRU PP(7,7)  STORED COLUMN WISE
C(350)
C(351) THRU C(366)      R(1,1) THRU R(4,4)  STORED COLUMN WISE
C(401) THRU C(428)      M(1,1) THRU M(4,7)  STORED COLUMN WISE
C(485)          PCNT
C(487)          DTCNT
C(488)          PGCNT
C(489)          ITCNT
C(490)          NORNDM
C(491) THRU C(495)      RNDMNO(1) THRU RNDMNO(5)  STORED COLUMN WISE
C(499)          NOLIST
C(500)          NOOUT
C(501) THRU C(550)      ONAME1(1) THRU ONAME1(50)  STORED COLUMN WISE
C(551) THRU C(600)      ONAME2(1) THRU ONAME2(50)  STORED COLUMN WISE
C(601) THRU C(650)      OUTNO(1)  THRU OUTNO(50)   STORED COLUMN WISE
C(651) THRU C(700)      LISTNO(1) THRU LISTNO(50)  STORED COLUMN WISE
C(701) THRU C(750)      VALUE(1)  THRU VALUE(50)   STORED COLUMN WISE
```

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

AIR FORCE INSTITUTE OF TECHNOLOGY (AFIT-SE)
WRIGHT-PATTERSON AFB, OHIO 45433

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

APPLICATION OF THE EXTENDED KALMAN FILTER TO BALLISTIC TRAJECTORY ESTIMATION AND PREDICTION

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

AFIT THESIS

5. AUTHOR(S) (First name, middle initial, last name)

JOSEPH C. ORNAT 1/LT USAF
DONALD K. POTTER CIV

6. REPORT DATE

June 1969

7a. TOTAL NO. OF PAGES

7b. NO. OF REFS

8a. CONTRACT OR GRANT NO.

b. PROJECT NO. N/A

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

GCC/EE/69-15

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of the Dean of Engineering, Air Force Institute of Technology (AFIT-SE), Wright-Patterson AFB, Ohio 45433.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

N/A

13. ABSTRACT

~~This~~ The thesis presents the results of a study wherein the Kalman filtering technique is applied to the estimation and prediction of the trajectory of a ballistic missile from radar measurements made from an airborne radar system. Any intercept system which is to guide an anti-missile is critically dependent on these computational functions.

The Kalman Filter equations are based on a number of assumptions that are not entirely justified in actual practice. For the case of estimating the state of ballistic re-entry vehicle on the basis of noisy measurements, the Kalman theory cannot be applied directly.

In ~~this~~ paper the Kalman estimator is extended to nonlinear trajectory equation and unknown ballistic parameters. An estimation and prediction model is developed assuming that azimuth, elevation, range and range-rate data is provided from a phased-array radar aboard an aircraft. In order to evaluate the model, a digital computer program was developed wherein a reference trajectory for a missile is generated and this information, along with tracker aircraft position, is used by a radar model to generated airborne tracking information which is contaminated with noise. From this information the Kalman estimation and prediction model yields estimates of the present states and future states of the target. These are compared with the reference trajectory to evaluate the model.

DD FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Kalman Filter						
Trajectory Estimation						
Radar Tracking						
Prediction						
Noise						
Equations of Motion						